Package 'ddR'

August 29, 2016

Title Distributed Data Structures in R

Version 0.1.2

Author Edward Ma, Indrajit Roy, Michael Lawrence

Maintainer Edward Ma <ema@hpe.com>

VignetteBuilder knitr

Description Provides distributed data structures and simplifies
distributed computing in R.

Depends methods, R (>= 3.0.0)

Imports parallel, Rcpp

Suggests testthat, Matrix, knitr

License GPL (>= 2) | file LICENSE

LazyData true

LinkingTo Rcpp

Collate 'RcppExports.R' 'ddR.R' 'dobject.R' 'ops.R' 'pdriver.R'
'pobject.R' 'zzz.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-11-25 08:54:21

R topics documented:

`as.darray` *Convert input matrix into a distributed array.*

Description

Convert input matrix into a distributed array.

Usage

```
as.darray(input, psize = NULL)
```

Arguments

`input`	input matrix that will be converted to darray.
`psize`	size of each partition as a vector specifying number of rows and columns.

Details

If partition size (psize) is missing then the input matrix is row partitioned and striped across the cluster, i.e., the returned distributed array has approximately as many partitions as the number of R instances in the session.

The last set of partitions may have fewer rows or columns if input matrix size is not an integer multiple of partition size. If 'A' is a 5x5 matrix, then 'as.darray(A, psize=c(2,5))' is a distributed array with three partitions. The first two partitions have two rows each but the last partition has only one row. All three partitions have five columns.

To create a distributed darray with just one partition, pass the dimension of the input frame, i.e. 'as.darray(A, psize=dim(A))'

Value

Returns a distributed array with dimensions equal to that of the input matrix and partitioned according to argument 'psize'. Data may reside as partitions on remote nodes.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

See Also

`darray` `psize`

Examples

```
## Not run:
##Create 4x4 matrix
mtx<-matrix(sample(0:1, 16, replace=T), nrow=4)
##Create distributed array spread across the cluster
da<-as.darray(mtx)
psize(da)
##Create distributed array with single partition
db<-as.darray(mtx, psize=dim(mtx))
psize(db)
##Create distributed array with two partitions
dc<- as.darray(mtx, psize=c(2,4))
psize(dc)
##Fetch first partition
collect(dc,1)

## End(Not run)
```

`as.dframe` *Convert input matrix or data.frame into a distributed data.frame.*

Description

Convert input matrix or data.frame into a distributed data.frame.

Usage

```
as.dframe(input, psize = NULL)
```

Arguments

`input`	input matrix or data.frame that will be converted to dframe.
`psize`	size of each partition as a vector specifying number of rows and columns.

Details

If partition size (psize) is missing then the input matrix/data.frame is row partitioned and striped across the cluster, i.e., the returned distributed frame has approximately as many partitions as the number of R instances in the session.

The last set of partitions may have fewer rows or columns if input matrix size is not an integer multiple of partition size. If 'A' is a 5x5 matrix, then 'as.dframe(A, psize=c(2,5))' is a distributed frame with three partitions. The first two partitions have two rows each but the last partition has only one row. All three partitions have five columns.

To create a distributed frame with just one partition, pass the dimension of the input frame, i.e. 'as.dframe(A, psize=dim(A))'

Value

Returns a distributed data.frame with dimensions equal to that of the input matrix and partitioned according to argument 'psize'. Data may reside as partitions on remote nodes.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

See Also

`dframe` `psize`

Examples

```
## Not run:
    ##Create 4x4 matrix
    mtx<-matrix(sample(0:1, 16, replace=T), nrow=4)
    ##Create distributed frame spread across the cluster
    df<-as.dframe(mtx)
    psize(df)
    ##Create distributed frame with single partition
    db<-as.dframe(mtx, psize=dim(mtx))
    psize(db)
    ##Create distributed frame with two partitions
    dc<- as.dframe(mtx, psize=c(2,4))
    psize(dc)
    ##Fetch first partition
    collect(dc,1)
    #creating of dframe with data.frame
    dfa <- c(2,3,4)
    dfb <- c("aa","bb","cc")
    dfc <- c(TRUE,FALSE,TRUE)
    df <- data.frame(dfa,dfb,dfc)
    #creating dframe from data.frame with default block size
    ddf <- as.dframe(df)
    collect(ddf)
    #creating dframe from data.frame with 1x1 block size
    ddf <- as.dframe(df,psize=c(1,1))
    collect(ddf)

## End(Not run)
```

as.dlist *Creates a distributed list from the input.*

Description

Creates a distributed list from the input.

Usage

```
as.dlist(items, nparts = NULL)

as.DList(items, nparts = NULL)
```

Arguments

items	The object to convert to a dlist.
nparts	The number of partitions in the resulting dlist.

Value

A dlist converted from the input. Note that a list of partitions (resulting from the use of parts()) may be used with as.dlist. This will recombine those partitions into a single distributed object.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

See Also

dlist

Examples

```
## Not run:
a <- as.dlist(list(1,2,3,4)) # A dlist with elements 1 to 4.
## A new dlist with only 2 partitions, which were partitions 3 and 4 of 'a'.
b <- as.dlist(parts(a,c(3,4)))

## End(Not run)
```

`cbind,DObject-method` *Column binds the objects.*

Description

Column binds the objects.

Usage

```
## S4 method for signature 'DObject'
cbind(..., deparse.level = 1)
```

Arguments

`...`	Objects to column bind.
`deparse.level`	Does nothing so far.

Value

A dobject with the operands (and their partitions) cbinded.

`collect` *Fetch partition(s) of 'darray', 'dframe' or 'dlist' from remote workers.*

Description

Fetch partition(s) of 'darray', 'dframe' or 'dlist' from remote workers.

Usage

```
collect(dobj, index = NULL)
```

Arguments

`dobj`	input distributed array, distributed data frame or distributed list.
`index`	a vector indicating partitions to fetch. If multiple indices are provided, the result is assembled in the same order as the indices provided, though be aware that for dframes and darrays the result may lose its structure.

Value

An R list, array, or data.frame containing data stored in the partitions of the input.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

Examples

```
## Not run:
a <- darray(dim=c(9,9),psize=c(3,3),data=5)
b <- collect(a) # 9x9 matrix filled with 5s
c <- collect(a,1) # First partition of a, which contains a 3x3 matrix of 5s

## End(Not run)
```

---

colMeans,DObject-method

Gets the column means for a distributed array or data.frame.

Description

Gets the column means for a distributed array or data.frame.

Usage

```
## S4 method for signature 'DObject'
colMeans(x, na.rm = FALSE, dims = 1L)
```

Arguments

x	The object to get the column means from.
na.rm	If TRUE, will remove NAs.
dims	Currently does nothing.

colnames,DObject-method

Gets the colnames for the distributed object.

Description

Gets the colnames for the distributed object.

Usage

```
## S4 method for signature 'DObject'
colnames(x)
```

Arguments

x	The distributed object to get the colnames for.

colSums,DObject-method

Get the column sums for a distributed array or data.frame.

Description

Get the column sums for a distributed array or data.frame.

Usage

```
## S4 method for signature 'DObject'
colSums(x, na.rm = FALSE, dims = 1L)
```

Arguments

x	The object to get the column sums from.
na.rm	If TRUE, will remove NAs.
dims	Currently does nothing.

`combine` *Combines a list of partitions into a single distributed object. (can be implemented by a frontend wrapper without actually combining data in storage).*

Description

Combines a list of partitions into a single distributed object. (can be implemented by a frontend wrapper without actually combining data in storage).

Usage

```
combine(driver, items)

## S4 method for signature 'ParallelddR,list'
combine(driver, items)
```

Arguments

`driver`	The driver on which combine is dispatched.
`items`	A list of partitions to combine.

Value

A new distributed object made form the items list.

`darray` *Creates a distributed array with the specified partitioning and contents.*

Description

Creates a distributed array with the specified partitioning and contents.

Usage

```
darray(nparts = NULL, dim = NULL, psize = NULL, data = 0,
  sparse = FALSE)

DArray(nparts = NULL, dim = NULL, psize = NULL, data = 0,
  sparse = FALSE)
```

Arguments

`nparts`	vector specifying number of partitions. If missing, 'psize' and 'dim' must be provided.
`dim`	the dim attribute for the array to be created. A vector specifying number of rows and columns.
`psize`	size of each partition as a vector specifying number of rows and columns. This parameter is provided together with dim.
`data`	initial value of all elements in array. Default is 0.
`sparse`	If TRUE, the output darray will be of type sparse_darray. The default value is FALSE.

Details

Array partitions are internally stored as dense matrices. Last set of partitions may have fewer rows or columns if the array size is not an integer multiple of partition size. For example, the distributed array 'darray(dim=c(5,5), psize=c(2,5))' has three partitions. The first two partitions have two rows each but the last partition has only one row. All three partitions have five columns.

Distributed arrays can also be defined by specifying just the number of partitions, but not their sizes. This flexibility is useful when the size of an array is not known apriori. For example, 'darray(nparts=c(5,1))' is a dense array with five partitions. Each partition can contain any number of rows, though the number of columns should be same to conform to a well formed array.

Distributed arrays can be fetched at the master using `collect`. Number of partitions can be obtained by `nparts`. Partitions are numbered from left to right, and then top to bottom, i.e., row major order. Dimension of each partition can be obtained using `psize`.

Value

Returns a distributed array with the specified dimensions. Data may reside as partitions in remote nodes.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

See Also

`collect` `psize` `dmapply`

Examples

```
## Not run:
## A 9 partition (each partition 3x3), 9x9 DArray with each element initialized to 5.
a <- darray(psize=c(3,3),dim=c(9,9),data=5)
collect(a)
b <- darray(psize=c(3,3),dim=c(9,9)) # Same as 'a', but filled with 0s.
## An empty darray with 6 partitions, 2 per column and 3 per row.
c <- darray(nparts=c(2,3))

## End(Not run)
```

---

ddR *Distributed Data-structures in R*

Description

ddR simplifies large-scale data analysis. It includes new language constructs to express distributed programs in R. Distributed programs writted in **ddR** can work across multiple execution engines such as **parallel**, **distributedR**, and others. **ddR** provides data-structures such as distributed array `darray` to partition and share data across multiple R instances. Users can express parallel execution using `dmapply`.

Commands

ddR contains the following commands. For more details use help function on each command.

Session manangement:

- `useBackend` - choose execution engine

Distributed array, data.frame, and list:

- `darray` - create distributed array
- `dframe` - create distributed data frame
- `dlist` - create distributed list
- `as.darray` - create darray object from matrix object
- `is.darray` - check if object is distributed array
- `parts` - obtain partitions of an object
- `nparts` - number of partitions as vector
- `totalParts` - obtain total number of partitions
- `psize` - obtain dimensions of partitions
- `collect` - fetch darray, dframe or dlist object at the master
- `repartition` - repartition input object

Distributed execution:

- `dmapply` - execute function on cluster
- `dlapply` - execute function on cluster

Author(s)

HP Vertica Development Team

References

- Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction (2015). _Sigmod 2015_, 1657-1668.
- Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. *EuroSys'13*, 197–210.
- Homepage: https://github.com/vertica/DistributedR

Examples

```
## Not run:
  library(ddR)
  useBackend(parallel)
  a <- dmapply(function(x,y) x+y, 1:5, 2:6, nparts=3)
  collect(a)

## End(Not run)
```

`ddRDriver-class` *The base S4 class for backend driver classes to extend.*

Description

The base S4 class for backend driver classes to extend.

Slots

`DListClass` A character vector naming the class-name for dlists.

`DArrayClass` A character vector naming the class-name for darrays.

`DFrameClass` A character vector naming the class-name for dframes.

`backendName` A character vector naming the backend.

dframe — *Creates a distributed data.frame with the specified partitioning and data.*

Description

Creates a distributed data.frame with the specified partitioning and data.

Usage

```
dframe(nparts = NULL, dim = NULL, psize = NULL, data = 0)

DFrame(nparts = NULL, dim = NULL, psize = NULL, data = 0)
```

Arguments

`nparts`	vector specifying number of partitions. If missing, 'psize' and 'dim' must be provided.
`dim`	the dim attribute for the data.frame to be created. A vector specifying number of rows and columns.
`psize`	size of each partition as a vector specifying number of rows and columns. This parameter is provided together with dim.
`data`	initial value of all elements in array. Default is 0.

Details

Data frame partitions are internally stored as data.frame objects. Last set of partitions may have fewer rows or columns if the dframe dimension is not an integer multiple of partition size. For example, the distributed data.frame 'dframe(dim=c(5,5), psize=c(2,5))' has three partitions. The first two partitions have two rows each but the last partition has only one row. All three partitions have five columns.

Distributed data.frames can also be defined by specifying just the number of partitions, but not their sizes. This flexibility is useful when the size of an dframe is not known apriori. For example, 'dframe(nparts=c(5,1))' is a dense array with five partitions. Each partition can contain any number of rows, though the number of columns should be same to conform to a well formed array.

Distributed data.frames can be fetched at the master using `collect`. Number of partitions can be obtained by `nparts`. Partitions are numbered from left to right, and then top to bottom, i.e., row major order. Dimension of each partition can be obtained using `psize`.

Value

Returns a distributed data.frame with the specified dimensions. Data may reside as partitions in remote nodes.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

See Also

`collect` `psize` `dmapply`

Examples

```
## Not run:
## A 9 partition (each partition 3x3), 9x9 dframe with each element initialized to 5.
a <- dframe(psize=c(3,3),dim=c(9,9),data=5)
collect(a)
b <- dframe(psize=c(3,3),dim=c(9,9)) # Same as 'a', but filled with 0s.
## An empty dframe with 6 partitions, 2 per column and 3 per row.
c <- dframe(nparts=c(2,3))

## End(Not run)
```

`dimnames,DObject-method`

Gets the dimnames for the distributed object.

Description

Gets the dimnames for the distributed object.

Usage

```
## S4 method for signature 'DObject'
dimnames(x)
```

Arguments

x	The distributed object to get the dimnames for.

`dimnames<-,DObject,list-method`

Sets the dimnames for the distributed object.

Description

Sets the dimnames for the distributed object.

Usage

```
## S4 replacement method for signature 'DObject,list'
dimnames(x) <- value
```

Arguments

`x`	The object to set the dimnames for.
`value`	The list of values, one vector per dimension, of names.

`dlapply`

Distributed version of 'lapply'. Similar to `dmapply`, but permits only one iterable argument, and output.type is always 'dlist'.

Description

Distributed version of 'lapply'. Similar to `dmapply`, but permits only one iterable argument, and output.type is always 'dlist'.

Usage

```
dlapply(X, FUN, ..., nparts = NULL)
```

Arguments

`X`	vector, matrix, list, data.frame, dlist, darray, or dframe or other iterable object to supply to the function in FUN.
`FUN`	the function to be applied to each element of 'X'.
`...`	optional arguments to 'FUN'.
`nparts`	number of partitions in the output dlist.

Value

a dlist with number of partitions specified in 'nparts'

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

Examples

```
## Not run:
a <- dlapply(1:5,function(x) x, nparts=3)
# A DList with 3 partitions,
# which in the aggregate contains the elements 1 through 5.
b <- dlapply(a,function(x) x+3) # AddR 3 to each element of dlist a.

## End(Not run)
```

`dlist` *Creates a distributed list with the specified partitioning and data.*

Description

Creates a distributed list with the specified partitioning and data.

Usage

```
dlist(..., nparts = NULL)

DList(..., nparts = NULL)
```

Arguments

`...`	values to initialize the dlist (optional).
`nparts`	number of partitions in the dlist. If NULL, nparts will equal the length of ...

Value

A dlist containing the data in ..., or an empty dlist, partitioned according to `nparts`.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

Examples

```
## Not run:
## A dlist containing 2 partitions, with data 1 to 4.
a <- dlist(1,2,3,4,nparts=2)
collect(a)

## End(Not run)
```

---

dmapply

Distributed version of mapply. Similar to R's 'mapply', it allows a multivariate function, FUN, to be applied to several inputs. Unlike standard mapply, it always returns a distributed object.

Description

Though dmapply is modeled after mapply, there are several important differences, as evident in the parameters described below.

Usage

```
dmapply(FUN, ..., MoreArgs = list(), output.type = c("dlist", "dframe",
  "darray", "sparse_darray"), nparts = NULL, combine = c("default", "c",
  "rbind", "cbind"))
```

Arguments

`FUN`	function to apply, found via 'match.fun'.
`...`	arguments to vectorize over (vectors or lists of strictly positive length, or all of zero length). These may also be distributed objects, such as dlists, darrays, and dframes.
`MoreArgs`	a list of other arguments to 'FUN'.
`output.type`	the output type of the distributed object. The default value of "dlist" means that the result of dmapply will be stored in a distributed list. "darray" will make dmapply return a darray, just as "dframe" will make it return a dframe. "sparse_darray" results in a special version of darray where the elements are sparse.
`nparts`	a 1d or 2d numeric value to specify how the output should be partitioned. dlists only have one-dimensional partitioning, whereas darrays and dframes have two (representing the number partitions across the vertical and horizontal dimensions).
`combine`	for dframes and darrays, it specifies how the results of dmapply are combined within each partition (if each partition contains more than one result). If "rbind", the results are stitched using rbind; if "cbind", cbind is used. If the value is "c", the results are flattened into one column, as is the case with simplify2array(). For dlists, "c" will first attempt to unlist each element of the dmapply result

and then expand these items within the partition of the dlist. One may think of this as the function that is invoked on the resulting list after the dmapply, with 'do.call'. The default value is "default", which for darrays and dframes has identical behavior to "c". For dlists, no function is called if "default".

Value

A dlist, darray, or dframe (depending on the value of output.type), with number of partitions equal to `nparts`

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

Examples

```
## Not run:
## A dlist created by adding two input vectors
a <- dmapply(function(x,y) x+y, 1:5, 2:6, nparts=3)
collect(a)

##Create a darray with 4 partitions. Partitions are stitched in 2x2 fashion,
# meaning the overall dims of the darray will be 4x4.
b <- dmapply(function(x) matrix(x,2,2), 1:4,output.type="darray",combine="rbind",nparts=c(2,2))
collect(b,1) #First partition
collect(b)

## End(Not run)
```

`DObject-class` *The baseline distributed object class to be extended by each backend driver. Backends may elect to extend once for all distributed object types ('dlist', 'darray', 'dframe,', etc.) for one per type, depending on needs.*

Description

The baseline distributed object class to be extended by each backend driver. Backends may elect to extend once for all distributed object types ('dlist', 'darray', 'dframe,', etc.) for one per type, depending on needs.

Slots

`nparts` Stores the 2d-partitioning scheme of the distributed object.

`psize` Stores, as a 2d-matrix (1d-for dlists) of the size of each partition.

`dim` The dimensions of the distributed object.

`backend` A character vector of the name of the backend that created the object.

`type` The distributed object type for this object (e.g,. 'dlist').

`do_collect` *Backend implemented function to move data from storage to the calling context (node).*

Description

Backend implemented function to move data from storage to the calling context (node).

Usage

```
do_collect(x, parts)

## S4 method for signature 'ParallelObj,integer'
do_collect(x, parts)
```

Arguments

`x`	The distributed object to fetch data from.
`parts`	The parts (indices) of the distributed object to fetch.

Value

The data returned as a list, matrix, or data.frame.

`do_dmapply` *Backend-specific dmapply logic. This is a required override for all backends to implement so dmapply works.*

Description

Backend-specific dmapply logic. This is a required override for all backends to implement so dmapply works.

Usage

```
do_dmapply(driver, func, ..., MoreArgs = list(), output.type = "dlist",
  nparts = NULL, combine = "default")

## S4 method for signature 'ParallelddR,`function`'
do_dmapply(driver, func, ...,
  MoreArgs = list(), output.type = c("dlist", "dframe", "darray",
  "sparse_darray"), nparts = NULL, combine = c("default", "c", "rbind",
  "cbind"))
```

Arguments

`driver`	The driver that the logic dispatches on.
`func`	The function to execute
`...`	Iterable arguments from dmapply.
`MoreArgs`	A list of more arguments to the funciton.
`output.type`	The type of output (can be 'dlist', 'darray', 'sparse_darray', or 'dframe').
`nparts`	A 2d-vector indicating how the output is partitioned.
`combine`	One of 'default', 'rbind', 'cbind', or 'c', which specifies how the results from each partition should be combined.

Value

An object specific to the backend, with the nparts and psize fields filled.

---

getBestOutputPartitioning

This is an overrideable function that determines what the output partitioning scheme of a dlapply or dmapply function should be. It determines the 'ideal' nparts for the output if it is not supplied. For API standard-enforcement, overriding this is not recommended.

Description

This is an overrideable function that determines what the output partitioning scheme of a dlapply or dmapply function should be. It determines the 'ideal' nparts for the output if it is not supplied. For API standard-enforcement, overriding this is not recommended.

Usage

```
getBestOutputPartitioning(driver, ..., nparts = NULL, type = NULL)

## S3 method for class 'ddRDriver'
getBestOutputPartitioning(driver, ..., nparts = NULL,
  type = NULL)
```

Arguments

`driver`	The backend driver to dispatch on.
`...`	The arguments to this dmapply operation.
`nparts`	The nparts argument, if any, supplied by the user.
`type`	The output.type supplied by the user.

Value

A 2d-vector, that will be passed into your backend's do_dmapply.

Methods (by class)

- `ddRDriver`: The default implementation for getBestOutputPartitioning.

`getPartitionIdsAndOffsets` *Gets the internal set of partitions, and offsets within each partition, of a set 1d or 2d-subset indices for a distributed object*

Description

It returns a list of 3 elements, where the first element is a list of partitions, the second is a list of row indices, and third a a list of column indices.

Usage

```
getPartitionIdsAndOffsets(indices, psizes, nparts)
```

Arguments

`indices`	A sorted list of sorted vectors, where the first element are the row indices, and the second (if 2d), column indices.
`psizes`	Partition-sizes matrix of the distributed object.
`nparts`	nparts vector of the distributed object.

Details

Note: This is an internal helper function of ddR.

get_parts *Gets the partitions to a distributed object, given an index.*

Description

Gets the partitions to a distributed object, given an index.

Usage

```
get_parts(x, index, ...)

## S4 method for signature 'ParallelObj,missing'
get_parts(x, index, ...)

## S4 method for signature 'ParallelObj,integer'
get_parts(x, index, ...)
```

Arguments

`x`	The distributed object to dispatch on.
`index`	The index or indices of the partitions to fetch.
`...`	Other options (not in use currently)

Value

A list containing the partitions of 'x'.

init *Called when the backend driver is initialized.*

Description

Called when the backend driver is initialized.

Usage

```
init(x, ...)

## S4 method for signature 'ddRDriver'
init(x, ...)

## S4 method for signature 'ParallelddR'
init(x, executors = NULL, type = "FORK", ...)
```

Arguments

`x`	The driver object to initialize the backend for.
`...`	Other parameters to pass to the initialization routine.
`executors`	Number of cores to run with.
`type`	If "FORK", will use UNIX fork() method. If "PSOCK", will use SNOW method.

Methods (by class)

- `ddRDriver`: Default backend initialization message.
- `ParallelddR`: Initialization for parallel

`is.darray` *Returns whether the input is a darray*

Description

Returns whether the input is a darray

Usage

```
is.darray(x)

is.DArray(x)
```

Arguments

`x`	input object.

Value

TRUE if x is a darray, FALSE otherwise.

Examples

```
## Not run:
is.darray(3) # FALSE
is.darray(darray(psize=c(3,3),dim=c(9,9))) # TRUE

## End(Not run)
```

`is.dframe` *Returns whether the input is a dframe*

Description

Returns whether the input is a dframe

Usage

```
is.dframe(x)

is.DFrame(x)
```

Arguments

`x`	input object.

Value

TRUE if x is a dframe, FALSE otherwise.

Examples

```
## Not run:
is.dframe(3) # FALSE
is.dframe(dframe(psize=c(3,3),dim=c(9,9))) # TRUE

## End(Not run)
```

`is.dlist` *Returns whether the input is a dlist*

Description

Returns whether the input is a dlist

Usage

```
is.dlist(x)

is.DList(x)
```

Arguments

`x`	Input object.

Value

TRUE if x is a dlist, FALSE otherwise

Examples

```
## Not run:
is.dlist(3) #FALSE
is.dlist(dlist(1,2,3,nparts=3)) #TRUE

## End(Not run)
```

`is.dobject` *Returns whether the input entity is a DObject*

Description

Returns whether the input entity is a DObject

Usage

```
is.dobject(x)

is.DObject(x)
```

Arguments

`x`	The input to test to see whether it is a DObject.

Value

TRUE if x is a DObject, FALSE otherwise

Examples

```
## Not run:
is.dobject(3) # FALSE
is.dobject(dlist(1,2,3,nparts=3)) # TRUE
is.dobject(darray(psize=c(3,3),dim=c(9,9))) # TRUE

## End(Not run)
```

`is.sparse_darray` *Returns whether the input is a sparse_darray*

Description

Returns whether the input is a sparse_darray

Usage

```
is.sparse_darray(x)
```

Arguments

`x`	input object.

Value

TRUE if x is a sparse_darray, FALSE otherwise.

Examples

```
## Not run:
is.sparse_darray(3) # FALSE
is.sparse_darray(darray(psize=c(3,3),dim=c(9,9))) # FALSE
is.sparse_darray(darray(npartitions=3,sparse=TRUE)) # TRUE

## End(Not run)
```

`mean,DObject-method` *Gets the mean value of the elements within the object.*

Description

Gets the mean value of the elements within the object.

Usage

```
## S4 method for signature 'DObject'
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

`x`	The distributed object to get the mean of.
`trim`	Not supported yet.
`na.rm`	If TRUE, removes NA values.
`...`	Other args.

names<-,DObject-method

Sets the names of a distributed object

Description

Sets the names of a distributed object

Usage

```
## S4 replacement method for signature 'DObject'
names(x) <- value
```

Arguments

`x`	The object whose names to set.
`value`	A vector with the names to set with.

nparts

Returns a 2d-vector denoting the number of partitions existing along each dimension of the distributed object, where the vector==c(partitions_per_column, partitions_per_row). For a dlist, the value is equivalent to c(totalParts(dobj),1).

Description

Returns a 2d-vector denoting the number of partitions existing along each dimension of the distributed object, where the vector==c(partitions_per_column, partitions_per_row). For a dlist, the value is equivalent to c(totalParts(dobj),1).

Usage

```
nparts(dobj)
```

Arguments

`dobj`	input distributed array, data.frame or list.

Value

A 2d-vector containing the number of partitions along each dimension.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

See Also

`totalParts`

Examples

```
## Not run:
a <- darray(psize=c(3,3),dim=c(9,9)) # 9 partitions of 3x3
b <- nparts(a) # returns c(3,3)

## End(Not run)
```

`parallel` *The default parallel driver*

Description

The default parallel driver

Usage

```
parallel
```

Format

```
Formal class 'ParallelddR' [package "ddR"] with 4 slots
  ..@ DListClass : chr "ParallelObj"
  ..@ DFrameClass: chr "ParallelObj"
  ..@ DArrayClass: chr "ParallelObj"
  ..@ backendName: chr "parallel"
```

Examples

```
## Not run:
useBackend(parallel,executors=4)

## End(Not run)
```

parts — *Retrieves, as a list of independent objects, pointers to each individual partition of the input.*

Description

Retrieves, as a list of independent objects, pointers to each individual partition of the input.

Usage

```
parts(dobj, index = NULL)
```

Arguments

`dobj`	input object.
`index`	numeric vector or list of indices referencing the partitions of the distributed object. If NULL, the returned list contains pointers to all partitions.

Details

parts() is primarily used in conjunction with dmapply when functions are written to be applied over partitions of distributed objects.

Value

a list of distributed objects, each referring to one partition of the input.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

Examples

```
## Not run:
a <- darray(psize=c(3,3),dim=c(9,9),data=3) # A darray of 9 partitions, each 3x3
b <- dmapply(function(x) sum(x), parts(a)) # dmapply to each 3x3 partition of 'a'
c <- parts(a,3) # A list containing one DObject, which is the 3rd partition of 'a'

## End(Not run)
```

psize *Return sizes of each partition of the input distributed object.*

Description

Return sizes of each partition of the input distributed object.

Usage

```
psize(dobj, index = NULL)
```

Arguments

`dobj`	input distributed object
`index`	a numeric vector or list containing the indices of the partitions. Default is NULL.

Value

A matrix that denotes the number of rows and columns in the partition. Row i of the matrix corresponds or size of i'th partition. For a dlist, the returned matrix has only 1 column.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

See Also

`nparts`, `parts`

Examples

```
## Not run:
a <- darray(psize=c(3,3),dim=c(9,9)) # 9 partitions of 3x3
b <- psize(a) # A 9x2 matrix, with each row containing c(3,3)

## End(Not run)
```

`rbind` *rbindddR*

Description

rbindddR

cbindddR

Usage

```
rbind(..., deparse.level = 1)

cbind(..., deparse.level = 1)
```

Arguments

`...`	objects to rbind or cbind
`deparse.level`	Does nothing so far.

Value

bound (cbind or rbind) dobject

`rbind,DObject-method` *row binds the arguments*

Description

row binds the arguments

Usage

```
## S4 method for signature 'DObject'
rbind(..., deparse.level = 1)
```

Arguments

`...`	Arguments to row bind.
`deparse.level`	Does nothing so far.

Value

A dobject with the opernads (and their partitions) rbinded.

`repartition`	*Repartitions a distributed object. This function takes two inputs, a distributed object and a skeleton. These inputs must both be distributed objects of the same type and same dimension. If 'dobj' and 'skeleton' have different internal partitioning, this function will return a new distributed object with the same internal data as in 'dobj' but with the partitioning scheme of 'skeleton'.*

Description

Repartitions a distributed object. This function takes two inputs, a distributed object and a skeleton. These inputs must both be distributed objects of the same type and same dimension. If 'dobj' and 'skeleton' have different internal partitioning, this function will return a new distributed object with the same internal data as in 'dobj' but with the partitioning scheme of 'skeleton'.

Usage

```
repartition(dobj, skeleton)

## S3 method for class 'DObject'
repartition(dobj, skeleton)
```

Arguments

`dobj`	distributed object whose data is to be preserved, but repartitioned.
`skeleton`	distributed Object whose partitioning is to be emulated in the output.

Value

A new distributed object with the data of 'dobj' and the partitioning of 'skeleton'.

Methods (by class)

- `DObject`: The default implementation of repartition.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

Examples

```
## Not run:
a <- dlist(1,2,3,4,nparts=2)
b <- dmapply(function(x) x, 11:14,nparts=4)
c <- repartition(a,b) # c will have 4 partitions of length 1 each, containing 1 to 4.

## End(Not run)
```

`rowMeans,DObject-method`

Gets the row means for a distributed array or data.frame.

Description

Gets the row means for a distributed array or data.frame.

Usage

```
## S4 method for signature 'DObject'
rowMeans(x, na.rm = FALSE, dims = 1L)
```

Arguments

x	The object to get the row means from.
na.rm	If TRUE, will remove NAs.
dims	Currently does nothing.

`rownames,DObject-method`

Gets the rownames for the distributed object.

Description

Gets the rownames for the distributed object.

Usage

```
## S4 method for signature 'DObject'
rownames(x)
```

Arguments

x	The distributed object to get the rownames for.

rowSums,DObject-method

Gets the row sums for a distributed array or data.frame.

Description

Gets the row sums for a distributed array or data.frame.

Usage

```
## S4 method for signature 'DObject'
rowSums(x, na.rm = FALSE, dims = 1L)
```

Arguments

`x`	The object to get the row sums from.
`na.rm`	If TRUE, will remove NAs.
`dims`	Currently does nothing.

shutdown

Called when the backend driver is shutdown.

Description

Called when the backend driver is shutdown.

Usage

```
shutdown(x)

## S4 method for signature 'ddRDriver'
shutdown(x)

## S4 method for signature 'ParallelddR'
shutdown(x)
```

Arguments

`x`	The driver object to shutdown.

Methods (by class)

- `ddRDriver`: Default backend shutdown message.
- `ParallelddR`: Shutdown for parallel

sum,DObject-method *Gets the sum of the objects.*

Description

Gets the sum of the objects.

Usage

```
## S4 method for signature 'DObject'
sum(x, ..., na.rm = FALSE)
```

Arguments

x	The first distributed object
...	Other objects
na.rm	If TRUE, removes the NA values.

totalParts *Returns the total number of partitions of the distributed object. The result is same as prod(nparts(dobj))*

Description

Returns the total number of partitions of the distributed object. The result is same as prod(nparts(dobj))

Usage

```
totalParts(dobj)
```

Arguments

dobj	input distributed array, data.frame, or list.

Value

The total number of partitions in the distributed object.

See Also

nparts

Examples

```
## Not run:
a <- darray(psize=c(3,3),dim=c(9,9)) # 9 partitions of 3x3
b <- totalParts(a) # Returns 9

## End(Not run)
```

`useBackend` *Sets the active backend driver. Functions exported by the 'ddR' package are dispatched to the backend driver. Backend-specific initialization parameters may be passed into the ellipsis (...) part of the function arguments.*

Description

The default driver uses R's 'parallel' as the backend.

Usage

```
useBackend(driver, ...)
```

Arguments

`driver`	driver object for the backend that will be used. This object should extend class 'ddRDriver', and the S4 methods for do_dmapply, do_collect, and get_parts should be defined in the class of the driver object.
`...`	additional parameters to pass to the initialization function of the driver.

Details

After successfully registering a new backend with useBackend(), all subsequent dmapply, collect, and parts operations will dispatch on that driver object's class. Note that distributed objects created with a different backend prior to switching will be incompatible with these backend-specific functions of the new driver.

References

Prasad, S., Fard, A., Gupta, V., Martinez, J., LeFevre, J., Xu, V., Hsu, M., Roy, I. Large scale predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. _Sigmod 2015_, 1657-1668.

Venkataraman, S., Bodzsar, E., Roy, I., AuYoung, A., and Schreiber, R. (2013) Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices. _EuroSys 2013_, 197-210.

Homepage: https://github.com/vertica/ddR

Examples

```
## Not run:
useBackend(parallel,executors=2)
library(distributedR.ddR); useBackend(distributedR)

## End(Not run)
```

`[` *Extract parts of a distributed object.*

Description

Extract parts of a distributed object.

Usage

```
## S4 method for signature 'DObject'
x[i, j,...,drop=TRUE]
```

Arguments

`x`	The distributed object to get the parts of.
`i`	The row index or indices to extract with.
`j`	The column index or indices to extract with.
`...`	Other args.
`drop`	If TRUE, vectorizable results will become vectors.

`[[,DObject,numeric-method`
Extracts a single element of a distributed object.

Description

Extracts a single element of a distributed object.

Usage

```
## S4 method for signature 'DObject,numeric'
x[[i, j, ...]]
```

Arguments

`x`	The object to get an element from.
`i`	The row index of the element.
`j`	The column index of the element.
`...`	Other args

`$,DObject-method` *Extracts elements of a distributed object matching the name.*

Description

Extracts elements of a distributed object matching the name.

Usage

```
## S4 method for signature 'DObject'
x$name
```

Arguments

`x`	The object to get the named element from.
`name`	The name vector to retrieve elements with.

Index