Package 'BiocParallel'

April 4, 2014

Type Package

Title Bioconductor facilities for parallel evaluation

Version 0.4.1

Author Martin Morgan and contributions from
Ryan Thompson <rct@thompsonclan.org>,Michel Lang <michellang@gmail.com>

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

Description This package provides modified versions and novel
implementation of functions for parallel evaluation, tailored to use with Bioconductor objects.

biocViews HighThroughputSequencing, Infrastructure

License GPL-2 | GPL-3

Imports methods, parallel, foreach, tools, BatchJobs, BBmisc

Suggests BiocGenerics, doParallel, RUnit

Collate.unix AllGenerics.R BiocParallelParam-class.R ErrorHandling.R
bpbackend-methods.R bpisup-methods.R bplapply-methods.R
bpmapply-methods.R bpschedule-methods.R bpstart-methods.R
bpstop-methods.R bpvec-methods.R bpvectorize-methods.R
bpworkers-methods.R bpaggregate-methods.R DoparParam-class.R
MulticoreParam-class.R pvectorize.R register.R
SerialParam-class.R SnowParam-class.R BatchJobsParam-class.R
utilities.R unix/mclapply.R unix/pvec.R unix/zzz.R

Collate.windows AllGenerics.R BiocParallelParam-class.R
ErrorHandling.R bpbackend-methods.R bpisup-methods.R
bplapply-methods.R bpmapply-methods.R bpschedule-methods.R
bpstart-methods.R bpstop-methods.R bpvec-methods.R
bpvectorize-methods.R bpworkers-methods.R bpaggregate-methods.R
DoparParam-class.R MulticoreParam-class.R pvectorize.R
register.R SerialParam-class.R SnowParam-class.R
BatchJobsParam-class.R utilities.R windows/zzz.R

R topics documented:

`BiocParallel-package` *Bioconductor facilities for parallel evaluation*

Description

This package provides modified versions and novel implementation of functions for parallel evaluation, tailored to use with Bioconductor objects.

Details

This package uses code from the `parallel` package,

Author(s)

Author: Martin Morgan and contributions from Ryan Thompson <rct@thompsonclan.org>, Michel Lang <michellang@gmail.com>

Maintainer: Bioconductor Package Maintainer <maintainer@bioconductor.org>

BatchJobsParam-class *Enable parallelization on batch systems*

Description

This class is used to parameterize scheduler options on managed high-performance computing clusters.

Usage

```
BatchJobsParam(workers, catch.errors = TRUE, cleanup = TRUE,
    work.dir = getwd(), stop.on.error = FALSE, seed = NULL,
    resources = NULL, conffile = NULL, cluster.functions = NULL,
    progressbar = TRUE, ...)
```

Arguments

workers	integer(1)Number of workers on with Multicore and SSH backend which defaults here to all workers available. On managed HPC workers defaults to NA but can be set to control chunking of jobs. See argument n.chunks in chunk and submitJobs for more information.
catch.errors	logical(1)Flag to determine in apply-like functions (see e.g. bplapply) whether to quit with an error as soon as one application fails or encapsulation of function calls in try blocks which triggers a resume mechanism (see bpresume). Defaults to TRUE.
cleanup	logical(1)BatchJobs creates temporary directories in the work.dir. If cleanup is set to TRUE (default), the directories are removed from the file systems automatically. Set this to FALSE whenever it might become necessary to utilize any special functionality provided by BatchJobs. To retrieve the registry, call loadRegistry on the temporary directory.
work.dir	character(1)Directory to store temporary files. Note that this must be shared across computational nodes if you use a distributed computing backend. Default ist the current working directory of R, see getwd.
stop.on.error	logical(1)Stop all jobs as soon as one jobs fails (stop.on.error == TRUE) or wait for all jobs to terminate. Default is FALSE.
seed	integer(1L)Set an initial seed for the RNG. See makeRegistry for more information. Default is NULL where a random seed is chosen upon initialization.
resources	list()List of job specific resources passed to submitJobs. Default is NULL where the resources defined in the configuration are used.
conffile	character(1)URI to a custom BatchJobs configuration file used for execution. Default is NULL which relies on BatchJobs to handle configuration files.
cluster.functions	ClusterFunctionsSpecify a specific cluster backend using on of the constructors provided by BatchJobs, see ClusterFunctions. Default is NULL where the default cluster functions defined in the configuration are used.

progressbar	logical(1) Suppress the progress bar used in BatchJobs and be less verbose. Default is FALSE.
...	Addition arguments, currently not handled.

BatchJobsParam constructor

Return an object with specified values. The object may be saved to disk or reused within a session.

Methods

The following generics are implemented and perform as documented on the corresponding help page: bpworkers, bpstart, bpstop, bpisup, bpbackend, bpbackend<-

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org

See Also

getClass("BiocParallelParam") for additional parameter classes.

register for registering parameter classes for use in parallel evaluation.

Examples

```
p <- BatchJobsParam()
bplapply(1:10, sqrt, BPPARAM=p)

## Not run:
register(BatchJobsParam(catch.errors = FALSE), default=TRUE)

## End(Not run)
```

BiocParallelParam-class

Virtual classes (for developer reference)

Description

These classes are primarily relevant to developers.

BiocParallelParam is a virtual class on which all parameter classes extend. It has no slots.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org

Examples

```
getClass("BiocParallelParam")
```

bpaggregate *Apply a function on subsets of data frames*

Description

This is a parallel version of aggregate.

Usage

```
## S4 method for signature formula,BiocParallelParam
bpaggregate(x, data, FUN, ..., BPPARAM)

## S4 method for signature data.frame,BiocParallelParam
bpaggregate(x, by, FUN, ..., simplify=TRUE, BPPARAM)

## S4 method for signature ANY,missing
bpaggregate(x, ..., BPPARAM)
```

Arguments

x	A data frame or a formula.
by	A list of factors. Data frame x will be split accoring to the factors.
data	A data frame.
FUN	Function to apply.
...	Additional arguments for FUN.
simplify	If set to TRUE, the return values of FUN will be simplified using simplify2array.
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation.

Value

See aggregate.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org.

bpcontrols *Control (start, stop, query) back-end Params*

Description

Use functions on this page to start, stop, and query the 'back-ends' that perform a parallel evaluation.

Usage

```
bpworkers(x, ...)

bpstart(x, ...)
bpstop(x, ...)
bpisup(x, ...)

bpbackend(x, ...)
bpbackend(x, ...) <- value
```

Arguments

x	An instance of a BiocParallelParam class, e.g., MulticoreParam, SnowParam, DoparParam. x can be missing, in which case the default back-end (see register) is used.
...	Additional arguments, perhaps used by methods.
value	Replace value for back-end.

Details

bpworkers reports the number of workers in the back-end as a scalar integer with value >= 0.

bpstart starts the back-end, if necessary. For instance, MulticoreParam back-ends do not need to be started, but SnowParam back-ends do. bp* functions like bplapply will automatically start the back-end if necessary.

bpstop stops the back-end, if necessary and possible.

bpisup tests whether the back-end is available for processing, returning a scalar logical value.

bpbackend retrieves an object representing the back end, if possible. Not all back-ends can be retrieved; see showMethods("backend").

bpbackend<- updates the back end, and is only meant for developer use.

Value

bpworkers returns a scalar integer >= 0.

bpisup returns a scalar logical.

bpstart, bpstop return an updated x, invisibly.

bpbackend, bpbackend<- return or accept back end-specific objects.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org.

See Also

BiocParallelParam for possible values of x.

Examples

```
bpworkers(SerialParam())

## Not run:
  p <- SnowParam(2L)
  bpworkers(p)          # 2 local nodes, communicating via sockets
  bpstart(p)            # start cluster
  bplapply(1:10, sqrt, BPPARAM=p)
  bpstop(p)             # stop cluster

  p <- SnowParam(4L)
  bplapply(1:10, sqrt, BPPARAM=p)  # automatically start / stop cluster

## End(Not run)
```

---

bplapply *Parallel lapply-like functionality*

Description

bplapply applies FUN to each element of X. Any type of object X is allowed, provided length, [, and [[methods are available. The return value is a list of length equal to X, as with lapply.

Usage

```
bplapply(X, FUN, ..., resume = getOption("BiocParallel.resume", FALSE),
    BPPARAM)

## S4 method for signature ANY,ANY
bplapply(X, FUN, ..., resume = getOption("BiocParallel.resume", FALSE),
    BPPARAM)

## S4 method for signature ANY,missing
bplapply(X, FUN, ..., resume = getOption("BiocParallel.resume", FALSE),
    BPPARAM)

## S4 method for signature ANY,BiocParallelParam
bplapply(X, FUN, ..., resume = getOption("BiocParallel.resume", FALSE),
    BPPARAM)
```

Arguments

X	Any object for which methods length, [, and [[are implemented.
FUN	The function to be applied to each element of X.
...	Additional arguments for FUN, as in lapply
resume	Flag to determine if a previous partially successful run should be resumed. See bpresume for details.
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation.

Details

When BPPARAM is missing, bplapply uses registered()[[1]], i.e., the default mechanism for parallel evaluation.

See showMethods{bplapply} for additional methods, e.g., method?bplapply("MulticoreParam").

Value

See lapply.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org. Original code as attributed in mclapply.

See Also

bpvec for parallel, vectorized calculations.

BiocParallelParam for possible values of BPPARAM.

Examples

```
showMethods("bplapply")

## ten tasks (1:10) so ten calls to FUN default registered parallel
## back-end. Compare with bpvec.
system.time(result <- bplapply(1:10, function(v) {
    message("working") ## 10 tasks
    sqrt(v)
}))
result
```

bpmapply *Parallel mapply-like functionality*

Description

bpmapply applies FUN to first elements of ..., the second elements and so on. Any type of object in ... is allowed, provided length, [, and [[methods are available. The return value is a list of length equal to the length of all objects provided, as with mapply.

Usage

```
bpmapply(FUN, ..., MoreArgs=NULL, SIMPLIFY=TRUE, USE.NAMES=TRUE,
    resume=getOption("BiocParallel.resume", FALSE), BPPARAM)

## S4 method for signature ANY,ANY
bpmapply(FUN, ..., MoreArgs=NULL, SIMPLIFY=TRUE, USE.NAMES=TRUE,
    resume=getOption("BiocParallel.resume", FALSE), BPPARAM)
## S4 method for signature ANY,missing
bpmapply(FUN, ..., MoreArgs=NULL, SIMPLIFY=TRUE, USE.NAMES=TRUE,
   resume=getOption("BiocParallel.resume", FALSE), BPPARAM)
## S4 method for signature ANY,BiocParallelParam
bpmapply(FUN, ..., MoreArgs=NULL, SIMPLIFY=TRUE, USE.NAMES=TRUE,
    resume=getOption("BiocParallel.resume", FALSE), BPPARAM)
```

Arguments

FUN	The function to be applied to each element passed via ... .
...	Objects for which methods length, [, and [[are implemented. All objects must have the same length or shorter objects will be replicated to have length equal to the longest.
MoreArgs	List of additional arguments to FUN.
SIMPLIFY	If TRUE the result will be simplified using simplify2array.
USE.NAMES	If TRUE the result will be named.
resume	Flag to determine if a previous partially successful run should be resumed. See bpresume for details.
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation.

Details

When BPPARAM is missing, bpmapply uses registered()[[1]], i.e., the default mechanism for parallel evaluation.

See showMethods{bplapply} for additional methods, e.g., method?bplapply("MulticoreParam").

Value

See lapply.

Author(s)

Michel Lang . Original code as attributed in mclapply.

See Also

bpvec for parallel, vectorized calculations.

BiocParallelParam for possible values of BPPARAM.

Examples

```
showMethods("bplapply")

## ten tasks (1:10) so ten calls to FUN default registered parallel
## back-end. Compare with bpvec.
system.time(result <- bplapply(1:10, function(v) {
    message("working") ## 10 tasks
    sqrt(v)
}))
result
```

---

bpresume *Resume computation with partial results*

Description

Resume partial successful calls to bplapply or bpmapply

Usage

```
bplasterror()
bpresume(expr)
```

Arguments

expr	expression A expression which calls either bplapply or bpmapply.

Usage

The resume mechanism is triggered if the argument `catch.errors` of the `BiocParallelParam` class is set to `TRUE`. The methods `bplapply` and `bpmapply` then store the current state of computation. Recalling the call directory with argument `resume` set to `TRUE` will then only compute the missing parts of the previous call and merge the results.

Alternatively, if the call to `bplapply` and `bpmapply` is inside a function not accessible directly by the user, the last call can be embedded into `bpresume` which sets an option accordingly to enable the resume feature down in the call stack.

The function `bplasterror` returns a `LastError` object containing the partial results and errors to investigate.

Note that nested calls of the apply functions can cause troubles in some scenarios.

Author(s)

Martin Morgan `mailto:mtmorgan@fhcrc.org`

---

bpschedule *Schedule back-end Params*

Description

Use functions on this page to influence scheduling of parallel processing.

Usage

```
bpschedule(x, ...)
```

Arguments

`x`	An instance of a `BiocParallelParam` class, e.g., `MulticoreParam`, `SnowParam`, `DoparParam`. `x` can be missing, in which case the default back-end (see `register`) is used.
`...`	Additional arguments, perhaps used by methods.

Details

`bpschedule` returns a logical(1) indicating whether the parallel evaluation should occur at this point.

Value

`bpschedule` returns a scalar logical.

Author(s)

Martin Morgan `mailto:mtmorgan@fhcrc.org`.

See Also

BiocParallelParam for possible values of x.

Examples

```
bpschedule(SnowParam())  # TRUE
bpschedule(MulticoreParam(2)) # FALSE on windows

p <- MulticoreParam(recursive=FALSE)
bpschedule(p)                          # TRUE
bplapply(1:2, function(i, p) {
    bpschedule(p)                      # FALSE
}, p = p, BPPARAM=p)
```

---

bpvec *Parallel, vectorized evaluation*

Description

bpvec applies FUN to subsets of X. Any type of object X is allowed, provided length, [, and c methods are available. The return value is a vector of length equal to X, as with FUN(X).

Usage

```
bpvec(X, FUN, ..., AGGREGATE=c, BPPARAM)

## S4 method for signature ANY,ANY
bpvec(X, FUN, ..., AGGREGATE=c, BPPARAM)

## S4 method for signature ANY,BiocParallelParam
bpvec(X, FUN, ..., AGGREGATE=c, BPPARAM)

## S4 method for signature ANY,missing
bpvec(X, FUN, ..., AGGREGATE=c, BPPARAM)
```

Arguments

X	Any object for which methods length, [, and c are implemented.
FUN	The function to be applied to subsets of X.
...	Additional arguments for FUN.
AGGREGATE	A function taking any number of arguments ... called to reduce results (elements of the ... argument of AGGREGATE from parallel jobs. The default, c, concatenates objects and is appropriate for vectors; rbind might be appropriate for data frames.
BPPARAM	A optional BiocParallelParam instance determining the parallel back-end to be used during evaluation.

Details

When BPPARAM is missing, bpvec uses registered()[[1]], i.e., the default mechanism for parallel evaluation.

When BPPARAM is an instance of a class derived from BiocParallelParam for which no other method applies, this method creates a vector of indexes and uses these in conjunction with bplapply to arrange for parallel evaluation.

When BPPARAM is a class for which no method is defined (e.g., SerialParam), FUN(X) is used.

See showMethods{bpvec} for additional methods, e.g., method?bpvec("MulticoreParam").

Value

The result should be identical to FUN(X, ...) (assuming that AGGREGATE is set appropriately).

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org. Original code as attributed in pvec.

See Also

bplapply for parallel lapply.

BiocParallelParam for possible values of BPPARAM.

Examples

```
showMethods("bpvec")

## ten tasks (1:10), called with as many back-end elements are specified
## by BPPARAM.  Compare with bplapply
system.time(result <- bpvec(1:10, function(v) {
    message("working") ## 10 tasks
    sqrt(v)
}))
result
```

---

bpvectorize *Transform vectorized functions into parallelized, vectorized function*

Description

This transforms a vectorized function into a parallel, vectorized function. Any function FUN can be used, provided its parallelized argument (by default, the first argument) has a length and [method defined, and the return value of FUN can be concatenated with c.

Usage

```
bpvectorize(FUN, ..., BPPARAM)

## S4 method for signature ANY,ANY
bpvectorize(FUN, ..., BPPARAM)

## S4 method for signature ANY,missing
bpvectorize(FUN, ..., BPPARAM)
```

Arguments

FUN	A function whose first argument has a length and can be subset [, and whose evaluation would benefit by splitting the argument into subsets, each one of which is independently transformed by FUN. The return value of FUN must support concatenation with c.
...	Additional arguments to parallization, unused.
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation.

Details

The result of bpvectorize is a function with signature ...; arguments to the returned function are the original arguments FUN. BPPARAM is used for parallel evaluation.

When BPPARAM is missing, bpvectorize uses registered()[[1]], i.e., the default mechanism for parallel evaluation.

When BPPARAM is a class for which no method is defined (e.g., SerialParam), FUN(X) is used.

See showMethods{bpvectorize} for additional methods, if any.

Value

A function taking the same arguments as FUN, but evaluated using bpvec for parallel evaluation across available cores.

Author(s)

Ryan Thompson mailto:rct@thompsonclan.org

See Also

bpvec

Examples

```
psqrt <- bpvectorize(sqrt) ## default parallelization
psqrt(1:10)
```

DoparParam-class *Enable parallel evaluation using registered dopar backend*

Description

This class is used to dispatch parallel operations to the dopar backend registered with the foreach package.

Usage

```
DoparParam(catch.errors = TRUE)
```

Arguments

catch.errors	logical(1)Flag to determine in apply-like functions (see e.g. bplapply) whether to quit with an error as soon as one application fails or encapsulation of function calls in try blocks which triggers a resume mechanism (see bpresume). Defaults to TRUE.

DoparParam constructor

Return a proxy object that dispatches parallel evaluation to the registered foreach parallel backend.

There are no options to the constructor. All configuration should be done through the normal interface to the foreach parallel backends.

Methods

The following generics are implemented and perform as documented on the corresponding help page (e.g., ?bpisup): bpworkers, bpstart, bpstop, bpisup, bpbackend, bpbackend<-, bpvec.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org

See Also

getClass("BiocParallelParam") for additional parameter classes.

register for registering parameter classes for use in parallel evaluation.

foreach-package for the parallel backend infrastructure used by this param class.

Examples

```
# First register a parallel backend with foreach
library(doParallel)
cl <- makeCluster(2)
registerDoParallel(cl)
```

```
p <- DoparParam()
bplapply(1:10, sqrt, BPPARAM=p)
bpvec(1:10, sqrt, BPPARAM=p)

stopCluster(cl)
## Not run:
register(DoparParam(), default=TRUE)

## End(Not run)
```

`MulticoreParam-class` *Enable multi-core parallel evaluation*

Description

This class is used to parameterize single computer multicore parallel evaluation on non-Windows computers.

Usage

```
MulticoreParam(workers = detectCores(), catch.errors = TRUE,
    setSeed = TRUE, recursive = TRUE, cleanup = TRUE,
    cleanupSignal = tools::SIGTERM, verbose = FALSE, ...)
```

Arguments

`workers`	`integer(1)`Number of workers on with Multicore and SSH backend which defaults here to all workers available. On managed HPC `workers` defaults to `NA` but can be set to control chunking of jobs. See argument `n.chunks` in `chunk` and `submitJobs` for more information.
`catch.errors`	`logical(1)`Flag to determine in apply-like functions (see e.g. `bplapply`) whether to quit with an error as soon as one application fails or encapsulation of function calls in `try` blocks which triggers a resume mechanism (see `bpresume`). Defaults to `TRUE`.
`setSeed`	`logical(1)`, as described in `parallel::mcparallel` argument `mc.set.seed`.
`recursive`	`logical(1)` indicating whether recursive calls are evaluated in parallel; see `parallel::mclapply` argument `mc.allow.recursive`.
`cleanup`	`logical(1)` indicating whether forked children will be terminated before `bplapply` returns, as for `parallel::mclapply` argument `cleanup`. If `TRUE`, then the signal sent to the child is `cleanupSignal`.
`cleanupSignal`	`integer(1)` the signal sent to forked processes when `cleanup=TRUE`.
`verbose`	`logical(1)` when `TRUE` echo stdout of forked processes. This is the complement of `parallel::mclapply`'s argument `mc.silent`.
`...`	Additional arguments passed to `mclapply`

MulticoreParam constructor

Return an object with specified values. The object may be saved to disk or reused within a session.

Methods

The following generics are implemented and perform as documented on the corresponding help page (e.g., ?bpisup): bpworkers, bpstart, bpstop, bpisup, bpschedule, bpbackend.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org

See Also

getClass("BiocParallelParam") for additional parameter classes.

register for registering parameter classes for use in parallel evaluation.

Examples

```
p <- MulticoreParam()
bplapply(1:10, sqrt, BPPARAM=p)
bpvec(1:10, sqrt, BPPARAM=p)

## Not run:
register(MulticoreParam(), default=TRUE)

## End(Not run)
```

---

register *Maintain a global registry of available back-end Params*

Description

Use functions on this page to add to or query a registry of back-ends, including the default for use when no BPPARAM object is provided to functions..

Usage

```
register(BPPARAM, default=TRUE)
registered(bpparamClass)
```

Arguments

BPPARAM	An instance of a BiocParallelParam class, e.g., MulticoreParam, SnowParam, DoparParam.
default	Make this the default BiocParallelParam for subsequent evaluations? If FALSE, the argument is placed at the lowest priority position.
bpparamClass	When present, the text name of the BiocParallelParam class (e.g., "MulticoreParam") to be retrieved from the registry. When absent, a list of all registered instances is returned.

Details

Registering a back-end provides a configuration for parallel evaluation. Only one instance of a particular BiocParallelParam class present in the registry. Except when default=FALSE, the most recently registered BiocParallelParam instance becomes the default for subsequent parallel evaluation.

Value

register returns, invisibly, a list of registered back-ends.

registered returns the back-end of type bpparamClass or, if bpparamClass is missing, a list of all registered back-ends.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org.

See Also

BiocParallelParam for possible values of BPPARAM.

Examples

```
registered()
```

SerialParam-class *Enable serial evaluation*

Description

This class is used to parameterize serial evaluation, primarily to facilitate easy transition from parallel to serial code.

Usage

```
SerialParam(catch.errors = TRUE)
```

Arguments

catch.errors	logical(1)Flag to determine in apply-like functions (see e.g. bplapply) whether to quit with an error as soon as one application fails or encapsulation of function calls in try blocks which triggers a resume mechanism (see bpresume). Defaults to TRUE.

SerialParam constructor

Return an object to be used for serial evaluation of otherwise parallel functions such as bplapply, bpvec.

Methods

The following generics are implemented and perform as documented on the corresponding help page (e.g., ?bpworkers): bpworkers. bpisup, bpstart, bpstop, are implemented, but do not have any side-effects.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org

See Also

getClass("BiocParallelParam") for additional parameter classes.

register for registering parameter classes for use in parallel evaluation.

Examples

```
p <- SerialParam()
simplify2array(bplapply(1:10, sqrt, BPPARAM=p))
bpvec(1:10, sqrt, BPPARAM=p)

## Not run:
register(SerialParam(), default=TRUE)

## End(Not run)
```

SnowParam-class *Enable simple network of workstations (SNOW) parallel evaluation*

Description

This class is used to parameterize simple network of workstations (SNOW) parallel evaluation on one or several physical computers.

Usage

```
SnowParam(workers = 0L, type, catch.errors = TRUE, ...)

## invoke as(cl, "SnowParam")
## S4 method for signature SOCKcluster,SnowParam
coerce(from, to)
```

Arguments

workers	integer(1)Number of workers on with Multicore and SSH backend which defaults here to all workers available. On managed HPC workers defaults to NA but can be set to control chunking of jobs. See argument n.chunks in chunk and submitJobs for more information.
type	character(1) type of cluster to use, as described in clusterMap argument type.
catch.errors	logical(1) Flag to determine in apply-like functions (see e.g. bplapply) whether to quit with an error as soon as one application fails or encapsulation of function calls in try blocks which triggers a resume mechanism (see bpresume). Defaults to TRUE.
...	Additional arguments passed to makeCluster
from, to	(N.B. Use as(from, "SnowParam") to coerce from a cluster created with, e.g., parallel::makeCluster). from is a SOCKcluster or derived instance (e.g., from parallel::makeCluster), to be coerced to a SnowParam instance.

SnowParam constructor

Return an object representing a SNOW cluster. The cluster is not created until bpstart is called.

bpstart creates the cluster by invoking makeCluster with arguments spec=workers, type, and other arguments passed to ... in SnowParam.

Use as(cl, "SnowParam") to coerce a cluster created directly by parallel::param to a SnowParam instance. Instances created in this way cannot be started or stopped.

Methods

The following generics are implemented and perform as documented on the corresponding help page (e.g., ?bpisup): bpworkers, bpstart, bpstop, bpisup, bpbackend, bpbackend<-, bpvec.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org

See Also

getClass("BiocParallelParam") for additional parameter classes.

register for registering parameter classes for use in parallel evaluation.

Examples

```
p <- SnowParam(2L)
bplapply(1:10, sqrt, BPPARAM=p)
bpvec(1:10, sqrt, BPPARAM=p)

## Not run:
register(SnowParam(2L), default=TRUE)

## End(Not run)
```

Index