Package 'BiocParallel'

April 4, 2013

Type Package

Title Bioconductor facilities for parallel evaluation

Version 0.2.0

Author Martin Morgan

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

Suggests BiocGenerics, doParallel

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

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

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")
```

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, ..., BPPARAM)

## S4 method for signature 'ANY,ANY'
bplapply(X, FUN, ..., BPPARAM)

## S4 method for signature 'ANY,missing'
bplapply(X, FUN, ..., 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
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.

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

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
```

---

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 of `c(FUN(X[i0]), FUN(X[i1]), ...)` where `X[i0]` etc., are sequential subsets of `X`.

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)
```

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(), setSeed = TRUE, recursive = TRUE,
    cleanup = TRUE, cleanupSignal = tools::SIGTERM, verbose = FALSE,
    ...)

## S4 method for signature 'ANY,MulticoreParam'
bplapply(X, FUN, ..., BPPARAM)

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

## S4 method for signature 'ANY,MulticoreParam'
bparallelize(X, ..., BPPARAM)
```

Arguments

X	A vector-like object supporting length, [, and, for bplapply, [[.
FUN	A function operating on X and other arguments in ....
...	Additional arguments to FUN, or for classes derived from MulticoreParam.
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	An MulticoreParam-class instance.
workers	integer(1) number of cores to use for parallel evaluation.
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.

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()
```

SerialParam constructor

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

The object is 'read-only'.

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, ...)

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

## S4 method for signature 'ANY,SnowParam'
bplapply(X, FUN, ..., BPPARAM)
```

Arguments

workers	Number of workers to start the cluster, as in makeCluster argument spec.
type	character(1) type of cluster to use, as described in parLapply argument type.
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.
X	A vector-like object supporting length, [, and, for bplapply, [[.
FUN	A function operating on X and other arguments in ....
...	Additional arguments to FUN, or for classes derived from MulticoreParam.
BPPARAM	An MulticoreParam-class 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