Package 'beachmat'

February 18, 2019

Version 1.4.0

Date 2018-10-22

Title Compiling Bioconductor to Handle Each Matrix Type

Encoding UTF-8

Depends R (>= 3.5)

Imports utils, methods, Rhdf5lib (>= 1.1.4), rhdf5, HDF5Array (>= 1.9.5), DelayedArray (>= 0.7.38), Rcpp (>= 0.12.14), BiocGenerics

Suggests testthat, BiocStyle, knitr, rmarkdown, Matrix, devtools

biocViews DataRepresentation, DataImport, Infrastructure

Description Provides a consistent C++ class interface for a variety of commonly used matrix types, including sparse and HDF5-backed matrices.

License GPL-3

NeedsCompilation yes

VignetteBuilder knitr

SystemRequirements C++11

LinkingTo Rcpp, Rhdf5lib

RoxygenNote 6.1.0

git_url https://git.bioconductor.org/packages/beachmat

git_branch RELEASE_3_8

git_last_commit e3b7a21

git_last_commit_date 2018-10-30

Date/Publication 2019-02-18

Author Aaron Lun [aut, cre], Hervé Pagès [aut], Mike Smith [aut]

Maintainer Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>

R topics documented:

`getBestChunkDims` *Get best chunk dimensions*

Description

Computes the optimal chunk dimensions for consecutive row/column access from a HDF5Matrix.

Usage

```
getBestChunkDims(dims)
```

Arguments

`dims`	An integer vector of length 2 containing the dimensions of a HDF5Matrix object.

Details

Consider a HDF5Matrix where access to consecutive rows or columns is requested. The optimal chunk dimensions ensure that the number of disk reads required is the same as that of a file layout with pure row or column chunks. This exploits the HDF5 chunk cache to store data from neighbouring rows/columns, avoiding the need to reread or rewrite the entire chunk for the next row/column. Obviously, this is not relevant to situations involving random row or column access.

Value

An integer vector of length 2, containing the dimensions of each chunk in the HDF5 file.

Author(s)

Aaron Lun

Examples

```
getBestChunkDims(c(10340, 234))
getBestChunkDims(c(13400, 2068))
```

`pkgconfig` *Compiler configuration arguments for use of beachmat*

Description

This function returns values for `PKG_LIBS` and `PKG_CPPFLAGS` variables for use in Makevars files. See `vignette("beachmat", "beachmat")` for details. Only `PKG_LIBS` should be needed in most cases. The environment variable `RBEACHMAT_RPATH` can be used to over-ride the inferred location of the installed package.

Usage

```
pkgconfig(opt = c("PKG_LIBS", "PKG_CPPFLAGS"))
```

Arguments

opt	A string specifying the compilation flags to print.

Value

Returns NULL and prints the corresponding value to stdout.

Author(s)

Aaron Lun

Examples

```
pkgconfig("PKG_LIBS")
pkgconfig("PKG_CPPFLAGS")
```

---

rechunkByMargins *Rechunk by margins*

Description

Convert an existing HDF5Matrix into a pure column- or row-based chunk layout.

Usage

```
rechunkByMargins(x, size=5000, outfile=NULL, outname=NULL,
    outlevel=NULL, byrow=TRUE)
```

Arguments

x	A HDF5Matrix object.
size	An integer scalar specifying the number of elements in each chunk.
outfile	A string containing the name for the output HDF5 file, chosen by getHDF5DumpFile if not specified.
outname	A string containing the name for the output HDF5 data set, chosen by getHDF5DumpName if not specified.
outlevel	An integer scalar specifying the compression level, chosen by getHDF5DumpCompressionLevel if not specified.
byrow	A logical scalar indicating if the output file should be row-chunked (default) or column-chunked.

Details

Pure column- or row-based chunk layouts are optimal for random column and row access, respectively, from a HDF5 file. This function can be used to convert a file into a pure row/column layout prior to calling other functions. In many cases, a small investment in rechunking time is repaid by a reduction in access times in downstream procedures.

Value

A HDF5Matrix object pointing to the HDF5 file containing the data from x but with the new chunk layout.

Author(s)

Aaron Lun

Examples

```
A <- as(matrix(runif(5000), nrow=100, ncol=50), "HDF5Array")
byrow <- rechunkByMargins(A, byrow=TRUE)
bycol <- rechunkByMargins(A, byrow=FALSE)
```

---

supportCppAccess *Support C++ access*

Description

Does the current matrix class support C++ access?

Usage

```
## S4 method for signature 'ANY'
supportCppAccess(x)
```

Arguments

x	A matrix-like object.

Details

This function is called by the **beachmat** C++ API upon encountering an unknown matrix type. If it returns TRUE, we assume that the package used to define the class of x also contains registered C++ functions to access rows or columns of x. This allows **beachmat** to use those C++ functions to directly access values of x, rather than relying on block realization of an unknown matrix.

If you need to use this function, you almost certainly need to read the vignette on "Extending beachmat".

Value

A logical scalar specifying whether C++ access is supported for x.

Author(s)

Aaron Lun

Examples

```
library(Matrix)
Y <- Matrix(0, 10, 10)
supportCppAccess(Y)

# Note that certain matrix types are always supported
# by beachmat but still return FALSE.
X <- matrix(0, 10, 10)
supportCppAccess(X)
```

Index