DelayedArray / HDF5Array update

Hervé Pagès

Fred Hutch, Seattle

April 2021

1. Recent additions to package DelayedArray

2. Recent additions to package HDF5Array

3. Work in progress and future work

1. Recent additions to package DelayedArray
2. Recent additions to package HDF5Array
3. Work in progress and future work

ConstantArray objects (by Aaron)

This would ordinarily take up 8 TB of memory:

```
library(DelayedArray)
CM <- ConstantArray(c(1e6, 1e6), value=NA_real_)
CM

## <1000000 x 1000000> ConstantMatrix object of type "double":
##                   [,1]       [,2]       [,3] ...  [,999999] [,1000000]
##       [1,]          NA         NA         NA   .          NA          NA
##       [2,]          NA         NA         NA   .          NA          NA
##       [3,]          NA         NA         NA   .          NA          NA
##       [4,]          NA         NA         NA   .          NA          NA
##       [5,]          NA         NA         NA   .          NA          NA
##        ...           .          .          .   .           .           .
##  [999996,]          NA         NA         NA   .          NA          NA
##  [999997,]          NA         NA         NA   .          NA          NA
##  [999998,]          NA         NA         NA   .          NA          NA
##  [999999,]          NA         NA         NA   .          NA          NA
## [1000000,]          NA         NA         NA   .          NA          NA

lobstr::obj_size(CM)

## 1.33 kB
```

sinkApply()

`sinkApply()`: a convenience function for walking on a `RealizationSink` derivative for the purpose of filling it with blocks of data

Example: Fill a 1e6 x 1e6 on-disk matrix with random data

```
sink <- HDF5RealizationSink(c(1e6L, 1e6L))  # or TileDBRealizationSink

sink_grid <- defaultSinkAutoGrid(sink)

FUN <- function(sink, viewport) {
    block <- array(runif(length(viewport)), dim=dim(viewport))
    write_block(sink, viewport, block)
}

sink <- sinkApply(sink, FUN, grid=sink_grid)

close(sink)
M <- as(sink, "DelayedArray")
```

rbind(), cbind(), and sparsity

`rbind()` and `cbind()` of `DelayedArray` objects now propagate sparsity

```
tenx1 <- HDF5Array::TENxMatrix("tenx1.h5")  # is_sparse(tenx1) is TRUE
tenx2 <- HDF5Array::TENxMatrix("tenx2.h5")  # is_sparse(tenx2) is TRUE

bigtenx <- cbind(tenx1, tenx2)
is_sparse(bigtenx)  # TRUE

blockApply(bigtenx, FUN, ...)   # will take advantage of sparsity
```

1. Recent additions to package DelayedArray

2. Recent additions to package HDF5Array

3. Work in progress and future work

Recent additions to package HDF5Array

`HDF5Array()`: can now take an URL to a file on Amazon S3 (kind of slow!)

`H5SparseMatrix`: a `DelayedMatrix` subclass for representing and operating on an HDF5 sparse matrix stored in CSR/CSC/Yale format (e.g. 10x Genomics and h5ad formats)

1. Recent additions to package DelayedArray

2. Recent additions to package HDF5Array

3. Work in progress and future work

Work in progress and future work

Work in progress:

`h5summarize(..., op="sum")`: Optimized summarization of an HDF5 dataset or subset:

- Implemented in C (direct calls to HDF5 C lib in *Rhdf5lib*)
- Operates at the level of the physical chunks
- More efficient than `blockApply()`
- Integration to *DelayedArray*/*DelayedMatrixStats*: `h5summarize()` will be used behind the scene by things like `rowVars()`

Future work:

`SparseArray` objects: In-memory sparse representation of arrays of arbitrary dimensions

- Already used internally by block processing of sparse `DelayedArray` objects (current name is `SparseArraySeed`)
- Will go to their own package (currently in `DelayedArray`)
- Implement fast native operations: arithmetic, `Math` group (e.g. `log`), summarization, etc.. This will benefit block processing of sparse `DelayedArray` objects