h5SE – experiments with delayed HDF5 export/import for SummarizedExperiments

Vincent J. Carey, stvjc at channing.harvard.edu

March 06, 2017

Contents

1 Introduction

We want to get some background on the use of HDF5 and DelayedArray infrastructure for modest-size assay sets.

We'll work with some 450k data.

```
suppressPackageStartupMessages({
library(SummarizedExperiment)
library(HDF5Array)
library(yriMulti)
if (!exists("banovichSE")) data(banovichSE)
})
banovichSE
## class: RangedSummarizedExperiment
## dim: 329469 64
## metadata(0):
## assays(1): betas
## rownames(329469): cg00000029 cg00000165 ... ch.9.98989607R
##   ch.9.991104F
## rowData names(10): addressA addressB ... probeEnd probeTarget
## colnames(64): NA18498 NA18499 ... NA18489 NA18909
## colData names(35): title geo_accession ... data_row_count naid
```

2 Export a standard RangedSummarizedExperiment

Let's use the newly defined method for exporting the assay component of a SummarizedExperiment:

```
library(benchOOM)
tf = tempfile()
dir.create(tf)
banoSE = saveHDF5SummarizedExperiment(banovichSE, tf, replace=TRUE)
banoSE
```

```
## class: RangedSummarizedExperiment
## dim: 329469 64
## metadata(0):
## assays(1): betas
## rownames(329469): cg00000029 cg00000165 ... ch.9.98989607R
##   ch.9.991104F
## rowData names(10): addressA addressB ... probeEnd probeTarget
## colnames(64): NA18498 NA18499 ... NA18489 NA18909
## colData names(35): title geo_accession ... data_row_count naid
```

Access capabilities:

```
assay(banoSE[1,])
## DelayedMatrix object of 1 x 64 doubles:
##               NA18498    NA18499    NA18501       .    NA18489    NA18909
## cg00000029  0.4733963  1.2943041 -0.8084735       .  0.6708168 -0.8609302
subsetByOverlaps(banoSE, rowRanges(banoSE)[1001:1010])
## class: RangedSummarizedExperiment
## dim: 10 64
## metadata(0):
## assays(1): betas
## rownames(10): cg00063006 cg00063040 ... cg00063346 cg00063357
## rowData names(10): addressA addressB ... probeEnd probeTarget
## colnames(64): NA18498 NA18499 ... NA18489 NA18909
## colData names(35): title geo_accession ... data_row_count naid
```

3 Import a packaged HDF5RangedSummarizedExperiment

```
litg = getLitGeu()
litg
## class: RangedSummarizedExperiment
## dim: 100 462
## metadata(3): MIAME constrHist colDataSource
## assays(1): exprs
## rownames(100): ENSG00000152931.6 ENSG00000183696.9 ...
##   ENSG00000239872.1 ENSG00000179023.6
## rowData names(18): source type ... tag ccdsid
## colnames(462): HG00096 HG00097 ... NA20826 NA20828
## colData names(35): Source.Name Comment.ENA_SAMPLE. ...
##   Factor.Value.laboratory. popcode
```

4 Benchmarking exercise

We want to create a spectrum of biologically meaningful queries. To this end we introduced the gobrowse and go2ranges functions.

```
library(Organism.dplyr)
if (file.exists("~/hg19.sqlite"))
  hg19 = src_organism(dbpath="~/hg19.sqlite") else {
  library(TxDb.Hsapiens.UCSC.hg19.knownGene)
  hg19 = src_organism(TxDb.Hsapiens.UCSC.hg19.knownGene)
  }
r1 = go2ranges("transmembrane receptor activity", hg19) # length 2302
## 'select()' returned 1:1 mapping between keys and columns
## Joining, by = "entrez"
length(r1)
## [1] 2302
library(microbenchmark)
# wrap in some kind of harness structure -- outputs that move towards visualization
# shiny interface -- user can pick pathway size or sort or not ... statistics
# would need to be able to interrupt -- that does kill the whole session?
microbenchmark(as.array(assay(subsetByOverlaps(banovichSE, r1[1:20]))), times=5)
## Unit: milliseconds
##                                                      expr      min       lq
##  as.array(assay(subsetByOverlaps(banovichSE, r1[1:20]))) 101.2796 115.5969
##      mean   median       uq      max neval
##  134.2498 116.7083 125.2028 212.4615     5
microbenchmark(as.array(assay(subsetByOverlaps(banoSE, r1[1:20]))), times=5)
## Unit: seconds
##                                                  expr      min       lq
##  as.array(assay(subsetByOverlaps(banoSE, r1[1:20]))) 1.141722 1.189497
##      mean   median       uq     max neval
##  1.450376 1.269488 1.359855 2.29132     5
# also include an exported, sorted GenomicRanges ... metadata work may be needed
```

This shows that the speed of getting a delayed array corresponding to a large pathway is similar if the base data are on disk in HDF5 or in memory.

However, applying as.array to get the results into memory seems slow for this particular request, and this could have something to do with the order of the rows as specified in the request. They may be scattered throughout the HDF5 store, and costly to recover together, in the order returned by the in memory request. Needs more investigation.

There is an indication that if the records requested are contiguous rows, the as.array moves quickly. . .