rhdf5client – HDF5 server access

Vincent J. Carey, stvjc at channing.harvard.edu, Shweta Gopaulakrishnan, reshg at channing.harvard.edu, Samuela Pollack, spollack at jimmy.harvard.edu

December 19, 2017

Contents

1 HDF5 server

HDF Server "extends the HDF5 data model to efficiently store large data objects (e.g. up to multi-TB data arrays) and access them over the web using a RESTful API." In this package, several data structures are introduced

- to model the server data architecture and
- to perform targeted extraction of numerical data from HDF5 arrays stored on the server.

We maintain, thanks to a grant from the National Cancer Institute, the server http://54.174.163.77:5000/. Visit this URL to get a flavor of the server structure: datasets, groups, and datatypes are high-level elements to be manipulated to work with data values from the server.

A key application of the rhdf5client package is support for the *restfulSE* package that defines an interface between the SummarizedExperiment class and the HDF5 Server. The server provides content for `assay()` requests to `RESTfulSummarizedExperiment` instances.

2 Some details

2.1 Motivation

Extensive human and computational effort is expended on downloading and managing large genomic data at site of analysis. Interoperable formats that are accessible via generic operations like those in RESTful APIs may help to improve cost-effectiveness of genome-scale analyses.

In this report we examine the use of HDF5 server as a back end for assay data.

A modest server configured to deliver HDF5 content via a RESTful API has been prepared and is used in this vignette.

2.2 Executive summary

We want to provide rapid read-only access to array-like data. To do this, the hierarchy and additional formalities of the HDF5 server data architecture are exposed through R functions and related classes. Full details on the HDF5 server are available at the HDFgroup site.

The dsmeta function returns top-level groups and datasets.

```
library(rhdf5client)
bigec2 = H5S_source("http://54.174.163.77:5000")
## analyzing groups for their links...
## done
bigec2
## HDF5 server domain:  http://54.174.163.77:5000
##  There are 10 groups.
##  Use groups(), links(), ..., to probe and access metadata.
##  Use dsmeta() to get information on datasets within groups.
##  Use [[ [dsname] ]]  to get a reference suitable for [i, j] subsetting.
dsmeta(bigec2)[1:2,]      # two groups
## DataFrame with 2 rows and 3 columns
```

```
##    groupnum                                dsnames
##   <integer>                                 <List>
## 1         1 tenx_400k_sorted,mike,neurons400k,...
## 2         2
##                               grp.uuid
##                             <character>
## 1 c3ca306c-3020-11e7-806d-123feca22a06
## 2 c3df8476-3020-11e7-806d-123feca22a06
dsmeta(bigec2)[1,2][[1]]  # all dataset candidates in group 1
## [1] "tenx_400k_sorted" "mike"             "neurons400k"
## [4] "tenx_full"        "tissues"          "assays"
## [7] "patelGBMSC"       "neurons100k"      "tenx_100k_sorted"
```

2.3 Hierarchy of server resources

2.3.1 Server

Given the URL of a server running HDF5 server, we create an instance of `H5S_source`:

```
mys = H5S_source(serverURL="http://54.174.163.77:5000")
## analyzing groups for their links...
## done
mys
## HDF5 server domain:  http://54.174.163.77:5000
##  There are 10 groups.
##  Use groups(), links(), ..., to probe and access metadata.
##  Use dsmeta() to get information on datasets within groups.
##  Use [[ [dsname] ]]  to get a reference suitable for [i, j] subsetting.
```

2.3.2 Groups

The server identifies a collection of 'groups'. For the server we are working with, only one group, at the root, is of interest.

```
groups(mys)
## DataFrame with 10 rows and 2 columns
##                                  groups    nlinks
##                                <character> <integer>
## 1  c3ca306c-3020-11e7-806d-123feca22a06        11
## 2  c3df8476-3020-11e7-806d-123feca22a06         1
## 3  c3ca7f5e-3020-11e7-806d-123feca22a06        88
## 4  c3cab2c6-3020-11e7-806d-123feca22a06         1
## 5  cbf54056-3020-11e7-806d-123feca22a06         4
## 6  c8f49ff0-3020-11e7-806d-123feca22a06        28
## 7  c3e00f40-3020-11e7-806d-123feca22a06         1
## 8  c3ca6640-3020-11e7-806d-123feca22a06         1
## 9  c3e20b6a-3020-11e7-806d-123feca22a06         1
## 10 c3ca9926-3020-11e7-806d-123feca22a06         3
```

2.3.3 Links for a group

There is a class to hold the link set for any group:

```
lks = links(mys,1)
lks
## HDF5 server link set for group c3ca306c-3020-11e7-806d-123feca22a06
##  There are 11 links.
##  Use targets([linkset]) to extract target URLs.
```

2.3.4 Dataset access

We use double-bracket subscripting to grab a reference to a dataset from an H5S source. A dataset must be two-dimensional, i.e., accessible with two subscripts.

```
dta = bigec2[["tenx_100k_sorted"]]
dta
## H5S_dataset instance:
##             dsname intl.dim1 intl.dim2              created     type.base
## 1 tenx_100k_sorted     1e+05     27998 2017-05-11T15:30:23Z H5T_STD_I32LE
```

2.3.5 Data

Data are accessed by subscripting. The subscripts are arrays of increasing, positive integers.

```
x = dta[ 15:20, 1905:1906 ]
x
##      [,1] [,2]
## [1,]   40    1
## [2,]   35    0
## [3,]   13    1
## [4,]  118    2
## [5,]   25    1
## [6,]   26    1
```

(Obsolete) The subscripts are colon-delimited character with initial, final and optional stride.

```
x = dta["15:20", "1904:1906"]
x
##      [,1] [,2]
## [1,]   35    0
## [2,]   13    1
## [3,]  118    2
## [4,]   25    1
## [5,]   26    1
```