BioHDF_Perl, a High Level Perl Package for Bioinformatics

User's Guide

The HDF Group
February, 2008

Summary. An explosion in the generation of sequence data has led to a search for new ways to handle that data. The HDF5 format, designed for managing scientific and engineering data, offers one component of a solution. Because the Perl programming language is heavily used in sequencing applications, a prototype Perl API called BioHDF_Perl has been developed to demonstrate how HDF5 could be made readily accessible to these applications.

This User's Guide briefly describes HDF5 and BioHDF_Perl, and how it can be used for storing DNA sequence data. It also describes how to install BioHDF_Perl, and gives an example of its use with FASTA sequence and quality data.

A companion BioHDF_Perl Reference Manual is available with details about the API reported here. For detailed information about HDF5 itself, see http://hdfgroup.org.

Introduction

The large amount of data present in biological systems and the need for its efficient processing and analysis have led to data intensive computing in the field of bioinformatics. Among the several applications to which this applies, DNA sequencing is of particular relevance.

For example, nucleic acid or peptide sequence data are currently typically stored and managed in text files using the FASTA format. Although this allows for convenient processing using scripting languages like Perl, as the number of sequences in a file increases, tasks like searching for a particular sequence can have large latencies. Also, the organization of complex data and metadata can be cumbersome when it must be stored and accessed strictly sequentially within a text file. Advanced file formats such as HDF5 can resolve these issues and provide for more efficient data management.

HDF5 is a widely used portable file format and library for storing, retrieving, analyzing, visualizing and exchanging data. HDF5 stores multidimensional arrays along with metadata in a portable file format. It supports hierarchical and other organizing structures, providing users with a high degree of flexibility for organizing and managing data. A high level API Perl package, BioHDF_Perl, has been developed specifically to facilitate the storage and management of genomic sequence data in HDF5 format.

BioHDF_Perl is a prototype designed to demonstrate and test the use of HDF5 for sequence data. Users of BioHDF_Perl are encouraged to comment on the usefulness of this prototype, as well as to suggest improvements and extensions.

This document describes the installation of BioHDF Perl, the structures generated in the HDF5 file, and the storage and retrieval operations. A sample script is used to illustrate the migration of sequence data from FASTA format into HDF5 files.

Installation

A prerequisite for installing BioHDF_Perl is the availability of binaries for the HDF5 1.6.5 library.

The Perl package can be installed using the following steps:

1. Modify Makefile.PL so that it points to the location of the HDF5 library.
2. perl Makefile.PL LIB=/install_path
3. make
4. make install

To use the Perl package, each script should contain the following lines at the beginning:

```
#!/usr/bin/perl

# load appropriate modules
use lib "/install_path";
use strict;
use Init;
use BioHDF_Perl;

# initialize HDF5 constants
Init::initialize();
```

These lines load the appropriate modules and initialize HDF5 constants necessary to create and access HDF5 files and datasets.

HDF5 Files and Datasets

The following is a brief introduction to HDF5. For more details, see the HDF5 documentation at http://www.hdfgroup.org/HDF5/doc/.

As suggested by the name Hierarchical Data Format, an HDF5 file can be hierarchically structured. The HDF5 groups and datasets implement this hierarchy.

In the simple and most common case, the file structure is a tree structure as shown in Figure 1; in the general case, the file structure may be a directed graph with a designated entry point. The tree structure is very similar to the file system structures employed on UNIX systems, directories and files, and on Apple Macintosh and Microsoft Windows systems, folders and files. HDF5 groups are analogous to the directories and folders; HDF5 datasets are analogous to the files. Thus, groups provide a way to organize objects within an HDF5 file, while datasets contain the application data.

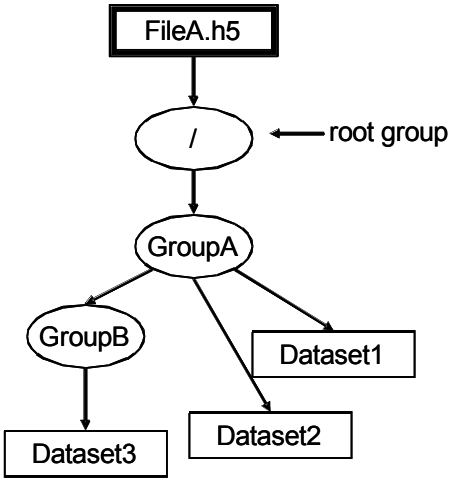

Figure 1 An HDF5 file with a strictly hierarchical group structure

An HDF5 dataset is an object composed of a collection of data elements, or raw data, and metadata that stores a description of the data elements, data layout, and all other information necessary to write, read, and interpret the stored data. From the viewpoint of the application the raw data is stored as a one-dimensional or multi-dimensional array of elements (the *raw data*), those elements can be of any of several numerical or character types, small arrays, or even compound types similar to database records.

A dataset may also include *attributes,* which are small metadata objects defined by applications to describe the nature and/or intended use of the dataset.

Storage of Genomic Sequence Data

One of the most utilized file formats in bioinformatics is the FASTA format. FASTA is a text-based file format used to describe nucleic acid or peptide sequences. A FASTA sequence record from the National Center for Biotechnology Information (NCBI) has the following form:

```
>gi|id1|gb|id2|id3 Comment about sequence
GATAATGGTA
```

A sequence record starts with a ">" followed by one or more identifiers for the particular sequence. After the first blank space, the rest of the line is considered to be a comment or description about the sequence. Actual sequence data starts in the next line and continues for as many lines as necessary. A companion "quality values" file is very similar but instead of bases, it contains quality values corresponding to the bases in the original file. In general, a FASTA file contains numerous sequence records.

BioHDF_Perl organizes sequence record data in an HDF5 file using a sequence collection structure, which consists of a group and four datasets: *ids, comments, sequences*, and *quals* as shown in Figure 2. Several sequence collections can be created in the same file.

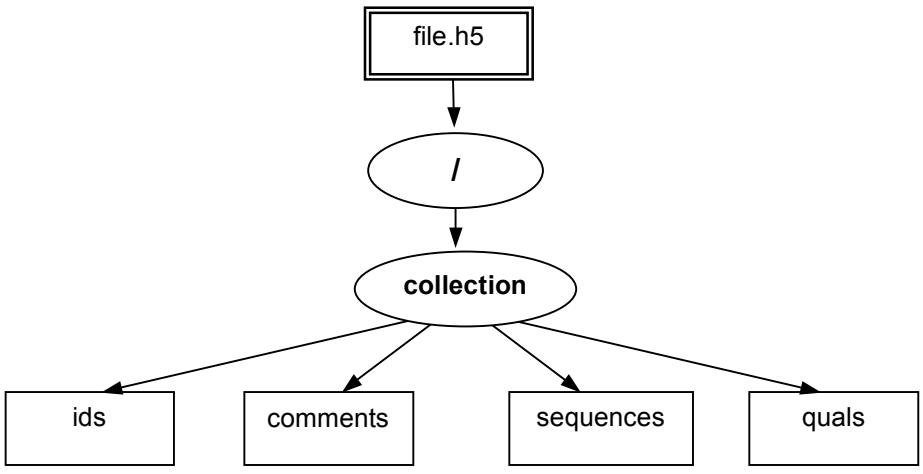

Figure 2 HDF5 structure for an sequence collection

The *sequences* and *quals* datasets are arrays that store the sequence bases and quality values, respectively. The *ids* and *comments* datasets are arrays of compound datatypes, which resemble database tables with the following description:

Dataset	Field	Description
ids	*id*	identifier for a particular sequence
	index	location of the sequence information in the *comments* dataset
comments	*comment*	description about the sequence
	offset	location of the sequence data in the *sequences* and *quals* datasets
	length	length of sequence data

The following FASTA records corresponding to a sample genomic sequence would be stored in an HDF5 file as shown in Figure 3.

```
>gi|id1|gb|id2|id3 Comment about sequence
GATAA

>gi|id1|gb|id2|id3 Comment about sequence
15 17 19 19 16
```

***IDS* DATASET**

array index	id	index
…	…	…
…	id1	i
…	id2	i
…	id3	i

***COMMENTS* DATASET**

array index	comment	offset	length
…	…	…	…
i	Comment about sequence	j	5
…	…	…	…

***SEQUENCES* DATASET**

array index	data element
…	…
j	G
j+1	A
j+2	T
j+3	A
j+4	A
…	…

***QUALS* DATASET**

array index	data element
…	…
j	15
j+1	17
j+2	19
j+3	19
j+4	16
…	…

Figure 3 Sequence record data in HDF5 file

The following steps are required to create an HDF5 file, a sequence collection, and to perform access to it:

```
File creation
Specify the file name.
Create the file.

      Collection creation
      Obtain the file identifier where the collection is to be
      created.
      Specify the collection name and description.
      Create the collection.

               Adding a sequence record into the collection
               Obtain the set of handles for the collection objects.
               Specify the sequence identifiers.
               Specify the sequence comment.
               Specify the sequence of bases.
               Specify the sequence of quality values.
               Perform the desired operation on the collection.

               Getting a sequence record from the collection
               Obtain the set of handles for the collection objects.
               Specify a sequence identifier.
               Perform the desired operation on the collection.

      Close the collection objects.

Close the file.
```

Programming Example

We will show the use of BioHDF_Perl APIs by means of a programming example. The following Perl script reads two FASTA files (downloaded from iFinch) containing sequence bases and quality values, respectively. An HDF5 file and a sequence collection are created using BioHDF_Perl APIs. Data from the FASTA records is parsed and stored in the HDF5 sequence collection. Sequence identifiers are sorted and a search is performed for a particular sequence.

```
#!/usr/bin/perl

use lib "/install_path";
use HDFPerl;
use strict;
use BioHDF_Perl;
use Init;

# initialize HDF5 environment
Init::initialize();

# input and output files
my $seq_filename = "FACTORVIII_01.fsa";
my $qual_filename = "FACTORVIII_01.qual";
my $hdf_filename = "FACTORVIII_01.h5";
```

```
# working vars
my $i=0;
my $ref;
my @header=();
my @ids=();
my @pre_ids=();

# create a new HDF File.
my $fid = BioHDF_Perl::create_sequence_file($hdf_filename);

# create sequence collection
my $collection = BioHDF_Perl::create_sequence_collection($fid,
      "collection1", "sequences from iFinch");

# open FASTA files
open(QUAL, "< $qual_filename");
my $line_qual = <QUAL>;

open(SEQ, "< $seq_filename");
my $line_seq = <SEQ>;

chomp $line_qual;
chomp $line_seq;

# iterate over every FASTA record
while ($line_qual){

    # sequence IDs is extracted from each sequence header line
    @header=split(/ /, $line_qual);
    @pre_ids=split(/>/, $header[0]);
    @ids=($pre_ids[1]);

    # rest of string is stored in $comments. We extract the comment
    # from the quality file because it is consistent with HDF5
    my $comment=join(' ',@header[1 .. $#header]);

    # working vars for iteration on each line of the current record
    # in both FASTA files (sequences and qualities)
    my $j=0;
    my @seq=();
    my @qual=();
    $qual[$j] = <QUAL>;
    $seq[$j] = <SEQ>;
    chomp $qual[$j];
    chomp $seq[$j];

    # iterate over every line of current FASTA record
    while ( ($qual[$j] !~ /^>/) && ($qual[$j]) ){
        $j++;
        $qual[$j] = <QUAL>;
        $seq[$j] = <SEQ>;
        chomp $qual[$j];
        chomp $seq[$j];
    }

    # quality data is arranged as an array of numbers
```

```
    my $quality = join(' ',@qual[0 .. $#qual-1]);
    my @post_quality = split(/ /, $quality);

    # sequence data is arranged as an array of bases. Arrays containing
    # base and quality values must have the same length.
    my $sequence = join('',@seq[0 .. $#seq-1]);

    my @post_sequence = ("");
    if (length($sequence) != 0) {
        @post_sequence = split(//, $sequence);
    }

    # add sequence into the collection
    BioHDF_Perl::add_sequence($collection, \@ids, $comment,
      \@post_sequence, \@post_quality);

    # settings for next iteration
    $i++;
    $line_qual=$qual[$#qual];
}
print "RECORDS READ\n$i\n\n";

# sort the sequence prior searching
BioHDF_Perl::sort_sequence_collection($collection);

# set ID of sequence to be found
my $key="FACTORVIII_01F_02.ab1";

# search sequence in the collection
my $ref = BioHDF_Perl::get_sequence($collection, $key);
print "ID\n$key\n\n";

# check whether sequence was found
if ($ref >= 0) {
    # dereference array
    my @seq=@{$ref};
    print "COMMENT\n$seq[0]\n\n";
    print "SEQUENCE\n@{$seq[1]}\n\n";
    print "QUALITIES\n@{$seq[2]}\n\n";
} else {
    print "$key not found \n\n";
}

# display collection description
my $description = BioHDF_Perl::get_collection_description($collection);
print "COLLECTION DESCRIPTION\n$description\n";

# close sequence collection and file
BioHDF_Perl::close_sequence_collection($collection);
BioHDF_Perl::close_sequence_file($fid);
```

List 1 Sample script showing basic HDF5 operations

The script starts with a few standard initial lines that set the environment, load the appropriate modules, initialize HDF5 constants, and define file names and working

variables. Then, the sequence file is created using the function `BioHDF_Perl::create_sequence_file`. This function specifies a filename and returns a file handle.
A sequence collection is created using the function `BioHDF_Perl::create_sequence_collection`. This function specifies a file handle, a collection name, and a description. It returns a reference for an array of handles for the dataset objects in the collection. Although it is possible to use a particular handle in the returned array, it is not necessary because the high level APIs use the reference for the entire array as an argument.

A while loop iterates over each sequence record in the FASTA file set. Data like identifiers and comments is parsed and stored in arrays and variables, respectively. A nested while loop goes over the lines corresponding to bases and quality values, which are placed into arrays of characters and integers.

Although the comments for the same genomic sequence are usually identical in a FASTA file set, there are cases in which they are not. One example is when the base sequence is empty but the quality sequence contains one value. The sequence length may be described in the comment in the FASTA files causing them to be different from each other. Since BioHDF_Perl makes sure that the arrays containing base and quality values have the same length, the script performs the following:

- When a base sequence is empty, the respective array contains one element, the null character, to match the length of the array containing the quality values.
- The sequence comment from the FASTA file containing the quality values is selected in this case because it is consistent with how data is stored in HDF5

Each sequence record is added to the collection using the function `BioHDF_Perl::add_sequence`. This function specifies a reference to the array of handles of the collection, a reference to the array containing sequence identifiers, a comment, and references to the arrays containing sequence bases and quality values.

Once the FASTA sequence records have been migrated into the HDF5 sequence collection, we can perform some management and retrieval operations. Before searching for a particular sequence in the collection using an identifier, it is recommended to sort the collection using the function `BioHDF_Perl::sort_sequence_collection`. If the number of records in the collection is very large, sorting the collection can improve the access performance significantly.

A particular record in the HDF5 collection can be accessed using the function `BioHDF_Perl::get_sequence`. This function specifies a reference to the array of handles of the collection, and the identifier of the desired sequence. If the sequence is found, it returns a reference to an array containing three elements: the sequence comment, a reference to an array of bases, and a reference to an array of quality values. Otherwise, it returns a negative value.

Finally, the HDF5 sequence collection and file are closed using the functions `BioHDF_Perl::close_sequence_collection` and `BioHDF_Perl::close_sequence_file`, respectively.

Remarks

- `BioHDF_Perl::create_sequence_file` creates an HDF5 file and returns the file identifier.

```
file_id HDFPerl::create_sequence_file(name)
```

  - The *name* parameter specifies the name of the file to be created.
  - This function returns the file identifier if successful, and a negative value otherwise.

- When a file is no longer accessed by a program, `BioHDF_Perl::close_sequence_file` must be called to release the resources used by the file. This call is mandatory.

```
status BioHDF_Perl::close_sequence_file(file_id)
```

- The root group is automatically created when a file is created. Every file has a root group and the path name of the root group is always `/`.

- `BioHDF_Perl::create_sequence_collection` creates the hierarchy shown in Figure 2, consisting of a group and four datasets in order to store a collection of sequences. It returns a reference to an array of handles for the objects in the hierarchy.

```
collection_ref
BioHDF_Perl::create_sequence_collection(file_id,
      name, description)
```

  - The *file_id* parameter specifies the file where the collection is to be created.
  - The *name* parameter specifies the name of the collection.
  - The *description* parameter specifies a description for the collection.
  - This function returns a reference to an array of handles for the objects in the collection if successful, and a negative value otherwise.

- When a collection is no longer accessed by a program, `BioHDF_Perl::close_sequence_collection` must be called to release the resources used by the collection. This call is mandatory.

```
status
BioHDF_Perl::close_sequence_collection(collection_ref)
```

- `BioHDF_Perl::get_collection_description` finds the description for a particular collection.

```
description BioHDF_Perl::get_collection_description(
      collection_ref)
```

  - The *collection* parameter is a reference to an array of handles for the collection.
  - The function returns the description for the collection if successful, and a negative value otherwise.

- `BioHDF_Perl::add_sequence` writes a sequence record into a collection.

```
status BioHDF_Perl::add_sequence(collection_ref, ids,
comment,
            bases, quals)
```

  - The *collection_ref* parameter is a reference to an array of handles for the objects in the collection.
  - The *ids* parameter is a reference to an array containing one or more identifiers for the sequence.
  - The *comment* parameter specifies the description for the sequence.
  - The *bases* parameter is a reference to an array containing the sequence bases.
  - The *quals* parameter is a reference to an array containing the sequence quality values.
  - The function returns a positive value if successful, and a negative value otherwise.

- `BioHDF_Perl::sort_sequence_collection` sorts the identifiers of the sequences in the collection.

```
status
BioHDF_Perl::sort_sequence_collection(collection_ref)
```

  - The *collection_ref* parameter is a reference to an array of handles for the objects in the collection.
  - The function returns a positive value if successful, and a negative value otherwise.

- `BioHDF_Perl::get_sequence` writes a sequence record into a collection.

```
sequence_ref BioHDF_Perl::get_sequence(collection_ref, id)
```

  - The *collection_ref* parameter is a reference to an array of handles for the objects in the collection.
  - The *id* parameter is the identifier for the desired sequence.

- If a sequence is found in the collection that corresponds to the given identifier, the function returns a reference to an array containing the following three elements:
    - The sequence comment.
    - A reference to an array containing the bases.
    - A reference to an array containing the quality values.
- Otherwise, it returns a negative value.