HDFql

the easy way to manage HDF5 data

This presentation was prepared by Rick (Mr. HDFql) who can't be here today and is presented by Gerd Heber (The HDF Group)

AGENDA

- What it is

- Who is using it

- How it works & examples

- Current features

- What's next!

WHAT IT IS

- HDFql stands for "Hierarchical Data Format query language" and is a high-level language to manage HDF5 data: *declarative*

- Designed to be as simple and powerful as SQL: adapted to HDF5

- Dramatically reduces the learning effort and time needed to handle HDF5: more time to focus on the problem at hand

- Reads and writes HDF5 data across multiple programming languages and platforms: a single *guest language* for multiple *host languages*

- Pre/post processes (HDF5) data using all nodes and cores available (and whenever possible): seamless *parallelism*

- Universities

- Research centers

- Biotech & renewable energy companies

- Automotive industry (electric vehicles)

- … you name it! ☺

HOW IT WORKS

- HDFql allows the execution of many operations to properly manage HDF5 (data). These are grouped in four categories:

  - <u>Data Definition Language (DDL):</u> create files, create groups, rename datasets, alter (i.e. extend) dimensions of datasets, copy attributes, …

  - <u>Data Manipulation Language (DML):</u> insert (i.e. write) data into datasets or attributes

  - <u>Data Query Language (DQL):</u> select (i.e. read) read data from datasets or attributes

  - <u>Data Introspection Language (DIL):</u> get group names, get dataset names (eventually stored in a certain group), get dimensions of attributes, …

HDFql

- *CREATE FILE my_file.h5*

  Create an HDF5 file named "my_file.h5"

- *CREATE FILE experiment.h5 IN PARALLEL*

  Create an HDF5 file named "experiment.h5" in parallel (i.e. using MPI)

- *CREATE GROUP countries*

  Create a group named "countries"

- *CREATE DATASET values AS FLOAT(20, 40) ENABLE ZLIB*

  Create a compressed dataset named "values" of data type float of two dimensions (size 20x40)

HOW IT WORKS (DML)

INSERT INTO my_dataset VALUES(3, 5, 7)

> Insert (i.e. write) values 3, 5 and 7 into dataset "my_dataset"

INSERT INTO measurements VALUES FROM EXCEL FILE values.xlsx

> Insert (i.e. write) values from Excel file "values.xlsx" into dataset "measurements"

INSERT DIRECTLY INTO raw VALUES(10, 20)

> Insert (i.e. write) values 10 and 20 directly (i.e. bypass several internal processing steps of the HDF5 library itself) into dataset "raw"

INSERT INTO dset(0:::1) VALUES FROM MEMORY 0

> Insert (i.e. write) values from a user-defined variable (that was previously registered and assigned to number 0) into the first chunk of dataset "dset" (using an hyperslab selection)

HDFql

Select (i.e. read) data from dataset "values" and populate cursor in use with it

- *SELECT FROM values*

Select (i.e. read) data from dataset "measurements" and write it into an Excel file "values.xlsx"

- *SELECT FROM measurements INTO EXCEL FILE values.xlsx*

Select (i.e. read) data directly (i.e. bypass several internal processing steps of the HDF5 library itself) from dataset "raw" and populate cursor in use with it

- *SELECT DIRECTLY FROM raw*

Select (i.e. read) 4th value of dataset "dset" (using a point selection) and write it into a user-defined variable (that was previously registered and assigned to number 0)

- *SELECT FROM dset(3) INTO MEMORY 0*

HDFql

- *SHOW*

> Show (i.e. get) all objects existing in current group

- *SHOW DATASET my_group/*

> Show (i.e. get) all datasets existing in group "my_group"

- *SHOW LIKE ***

> Show (i.e. get) all objects recursively starting from current group

- *SHOW ATTRIBUTE group2 LIKE **/1|3*

> Show (i.e. get) all attributes recursively starting from group "group2" that contain "1" or "3" in their names

CURRENT FEATURES

- Portable across programming languages (C, C++, Java, Python, C#, Fortran and R) and platforms (Windows, Linux and macOS)

- Supports direct chunk write and read

- Supports both point and (irregular) hyperslab selections

- Supports parallel HDF5 (i.e. HDF5 + MPI)

- Supports reading data from a text, binary or Excel file and writing it into an HDF5 dataset/attribute (and vice-versa)

- And many more...

WHAT'S NEXT

- Support HDF5 library version 1.10.x

- Support virtual datasets (VDS)

- Support single-write multiple-readers (SWMR)

- Support dynamically loaded filters

- Support HDF5 data stored in Amazon E3 and Hadoop Distributed File System (HDFS)

- Support additional programming languages (e.g. Go, Rust, JavaScript) and tools (e.g. MATLAB) through proper wrappers

HDFql

- Support pre/post processing data via pre-defined functions (using all nodes & cores available whenever possible) and user-defined functions.

  – INSERT INTO dset0 **MIN**(VALUES) FROM FILE input.txt   call pre-defined function "MIN" to return the minimum value stored in text file "input.txt" and write the value into object "dset0"

  – SELECT FROM **AVG**(dset1) IN PARALLEL   call pre-defined function "AVG" to return the average value of the data stored in object "dset1" in parallel (i.e. using MPI)

  – SELECT FROM **DUMMY**(dset2)   load shared library "HDFqlDUMMY.so" (by HDFql) and call user-defined function "DUMMY" to process the data stored in object "dset2" and return the result

- Support reading data from an HDF5 dataset/attribute and writing it into another HDF5 dataset/attribute (eventually using a function to pre/post process the data *en passant*).

  – INSERT INTO upper_names UPPER(VALUES) **FROM names**   call pre-defined function "UPPER" to convert values stored in object "names" in upper-case and write the result into object "upper_names"

  – SELECT FROM AVG(dset) **INTO ATTRIBUTE dset/average**   call pre-defined function "AVG" to return the average value of the data stored in object "dset" and write the value into attribute "average" attached to object "dset" (HDFql creates the attribute if it does not exist)

ADDITIONAL INFORMATION

- HDFql can be used both in commercial and non-commercial products, and completely free of charge

- Discuss on https://forum.hdfgroup.org/c/hdf-tools

- Support inquiries: https://www.hdfql.com/#contact

- All versions of HDFql ever publicly released are available at https://www.hdfql.com/releases

- Each released version of HDFql contains:
  - Libraries and wrappers for the C, C++, Java, Python, C#, Fortran and R programming languages

  - Examples that illustrate how to use HDFql in these programming languages

  - A command-line interface tool named "HDFqlCLI"

  - A complete reference manual describing HDFql and its operations

HDFql

the easy way to manage HDF5 data

New users, partnerships or potential sponsors always welcome

www.hdfql.com