HDFql

the easy way to manage HDF5 data

THE PROBLEM

- HDF5 is a very flexible and powerful data format that is heavily used in science & engineering – but most of its APIs are rather complex and difficult

- The HDF5 C library (i.e. the reference API) already has more than 400 functions and each new release of the library adds new functions!

- This puts tremendous technical challenges on users (who are typically not experts on data format/storage technologies) and ultimately slows down science and data driven innovation

THE SOLUTION (HDFql)

- HDFql stands for “Hierarchical Data Format query language” and is a high-level (declarative) language to manage HDF5 data

- It is designed to be as simple and powerful as SQL – and dramatically reduces the learning effort and time needed to handle HDF5

- By being a declarative language, users just need to “tell” HDFql **what** they want to achieve and it takes care of satisfying requests by dealing with all HDF5 low-level details – in contrast to imperative languages (which is the case of all existing HDF5 APIs) where users have to specify **how** their requests are handled through a lot of programming and knowledge of HDF5 low-level details!

WHO IS USING HDFql

- Universities
- Research centers
- Biotech companies
- Renewable energy organizations
- Auto industry (electrical vehicles)
- … you name it! :)

HOW HDFql WORKS

- HDFql allows the execution of many operations to properly manage HDF5 (data) in a declarative fashion. These are grouped in four categories:
  - Data Definition Language (DDL): create HDF5 files, create groups, rename datasets, alter (i.e. extend) dimensions of datasets, copy attributes, …
  - Data Manipulation Language (DML): insert (i.e. write) data into datasets or attributes
  - Data Query Language (DQL): select (i.e. read) read data from datasets or attributes
  - Data Introspection Language (DIL): get group names, get dataset names (eventually stored in a certain group), get dimensions of attributes, …

HOW HDFql WORKS (DDL)

- *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 HDFql 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)

HOW HDFql WORKS (DQL)

- *SELECT FROM values*

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

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

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

- *SELECT DIRECTLY FROM raw*

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 FROM dset(3) INTO MEMORY 0*

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)

HOW HDFql WORKS (DIL)

- *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 (IN HDFql 2.4.0)

- Supports disparate 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 both serial and 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 COMING IN HDFql 2.5.0

- Expected in Q4 2022
- Support HDF5 library version 1.8.23
- 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 HDF5 object "dset0"
  - SELECT FROM ***AVG***(dset1) IN PARALLEL **===>** call pre-defined function "AVG" to calculate the average value of the data stored in HDF5 object "dset1" in parallel (i.e. using MPI) and return the value to the user
  - SELECT FROM ***DUMMY***(dset2) **===>** load shared library "HDFqlDUMMY.so" (dynamically by HDFql) and call user-defined function "DUMMY" to process the data stored in HDF5 object "dset2" and return the result (of the processing) to the user

WHAT'S COMING IN HDFql 2.5.0 (cont.)

- Support sliding cursors to enable reading a dataset that does not fit in main memory (RAM) in a sliding fashion through a cursor, allowing a user to (transparently) load/process the dataset in an out-of-core manner

- Support automatic allocation of the necessary amount of memory associated to a user-defined variable based on the type and size of the data to store (alleviating the user from doing this allocation, which may be cumbersome)

- Support reading/writing a variable-length dataset/attribute into/from a user-defined variable in Java (through the *ArrayList* class) and C# (through the *List* class)

- Support the Go programming language through a proper wrapper

WHAT'S COMING IN HDFql 3.0.0

- Expected in Q2 2023
- 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 S3
- Support MATLAB (environment) through a proper wrapper

ADDITIONAL INFORMATION

- HDFql is completely free of charge and can be used both in commercial and non-commercial products without any restrictions
- Support can be provided by going at 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 C, C++, Java, Python, C#, Fortran and R
  - 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