Installation Guide – FastQuery

I. Overview

FastQuery provides a simple API for indexing and querying scientific format datasets using the FastBit bitmap index technology.

Datasets can be retrieved using complex compound range queries such as "(energy > 100) && (70 < pressure <90)". The bitmap index technology only retrieves the data elements that satisfy the query condition and shows significant speedup compared with reading the entire datasets.

* Current support file format:
* HDF5
* NetCDF (only supported in serialized mode))
* BP (under testing)
* Binary (under testing)
* Cache in memory (under testing)
* Current support data type:
* double
* float
* byte:char
* int:int32\_t
* unsigned int:uint32\_t
* long:int64\_t
* unsigned long:uint64\_t
* Source Code Structure

--src---const.h: constant header file

|--fastquery-config.h: header file generated from config

|--FlexLexer.h: header file for flex and lexer

|--fileDriver.h: unified I/O interface for array model data

|--fq.h: fastquery basic API

|--indexBuilder.h: fastquery index builder API

|--queryProcessor.h: fastquery query processor API

|--metadataMgr.h: manage and provide variable metadata information to fastquery

|--fqParser.h: fastquery parser object

|--fqPart: wrapper for the ibis::part object

|--fqColumn: wrapper for the ibis::column object

|--fqVar: data structure for storing variable information

|--fqIndex: wrapper for the ibis::rlic and ibis::bin objects

|--hdf5file.h: hdf5 implementation for the array I/O interface

|--queryParser.yy: define query syntax

|--queryLexer.ll: define query token

|--netCDFfile.h: hdf5 implementation for the array I/O interface

II. Required libraries

1. MPI

2. HDF5 (& Zlib)

3. FastBit

4. FastQuery0

III. Installation - Can only be built by **GNU** compiler

1. MPI

2. HDF5 - should be built for back end compute node

Take Mira as an example,

./configure CC=mpicc \

CFLAGS="-O3 -qnohot -g \

-DIBMR2Fortran \

-D\_LARGEFILE64\_SOURCE \

-D\_FILE\_OFFSET\_BITS=64 \

-D\_AIX -UH5\_HAVE\_GETPWUID" \

--prefix="$INSTALL\_PATH" \

--enable-parallel \

--enable-production \

--enable-debug=all \

--enable-using-memchecker \

(--with-zlib="$ZLIB\_PATH")

make -j4

make install

3. FastBit

Take Mira as an example,

./configure --enable-static \

--prefix="$INSTALL\_PATH" \

make LDFLAGS=-all-static -j4

make install

4. FastQuery

Take Mira as an example,

./configure CXX=mpicxx \

--disable-shared \

--disable-debug \

--no-create \

--no-recursion \

--prefix="$INSTALL\_PATH" \

--with-fastbit="$FASTBIT\_PATH" \

--with-hdf5="$HDF5\_PATH" \

(LDFLAGS=-L$Zlib\_PATH/lib)

make LDFLAGS=-all-static -j4

make install

make examples

IV. Examples

1. build indexes

mpirun –n $nproc $FQ\_PATH/examples buildIndexMPI \

-f $INPUT\_PATH

-i $OUTPUT\_INDEX\_PATH

-g $LOG\_PATH

-n VARIABLE NAME

-p VARIABLE PATH

-v VERBOSENESS

-m FILE MODEL {HDF5, NETCDF, BINARY, DSM}

-l MPI\_SUBARRAY\_SIZE

e.g. There is a hdf5 files stored in /home/user/input.h5

and has one dataset /Step#1/Energy

mpirun –n 100 /home/user/fq/examples buildIndexMPI \

-f /home/user/input.h5

-i /home/user/index.h5

-g /home/user/logfiles

-n "Energy[0:100000]"

-p "/Step#0"

-v 0

-m HDF5

-l 1000

2. query

mpirun –n $nproc $FQ\_PATH/examples buildIndexMPI \

-f $INPUT\_PATH

-i $OUTPUT\_INDEX\_PATH

-g $LOG\_PATH

-n VARIABLE NAME

-p VARIABLE PATH

-m FILE MODEL {HDF5, NETCDF, BINARY, DSM}

-v VERBOSENESS

-l MPI\_SUBARRAY\_SIZE

-q QUERY CONSTRAINT

e.g. There are a hdf5 files stored in /home/user/input.h5

with one dataset /Step#1/Energy,

and a index file in /home/user/index.h5

mpirun –n 100 /home/user/fq/examples buildIndexMPI \

-f /home/user/input.h5

-i /home/user/index.h5

-g /home/user/logfiles

-n "Energy[0:100000]"

-p "/Step#0"

-v 3

-m HDF5

-l 1000

-q "0.5 < Energy < 1.2"