Compression Plugins in h5wasm

Reading/writing compressed HDF5 on the web

Brian B. Maranville

NIST Center for Neutron Research

HDF User Group Meeting 2023-09-19

NIST CENTER FOR NEUTRON RESEARCH

Motivation: NCNR use case

- 655,778+ NeXus datafiles in existing public HTTP repository
- Desire for zero-install viewer/exploration tool
- Desire for viewing user-generated files
  - User-owned HDF5 (processed) outputs from NeXus inputs
  - Use same viewer for raw and processed data
- Required traceability:
  - Individual files with separately stored hash (SHA-256)
  - DOI for resolving file folders

NIST CENTER FOR NEUTRON RESEARCH

Chosen solution: browser-native viewer

- Backend (HDF5 reading library)
  - jsfive (pure javascript port of pyfive)
    - Direct implementation from HDF5 spec
    - Limited coverage of specification – hard to expand
  - h5wasm (Emcscripten-compiled high-level library)
    - Built against full HDF5 C API
- Frontend (plotting, tree browser)
  - Started with in-house viewer (jQuery + D3 + jsfive)
  - Moving to PaNOSC h5web https://h5web.panosc.eu

NIST CENTER FOR NEUTRON RESEARCH

h5wasm: JS/WASM library for HDF5

1. High-level utilities developed in TypeScript
   - Open File objects in read, append or write mode
   - Read or create groups, datasets and attributes
   - Modeled after (but much more limited than) h5py
   - Calls low-level functions from webassembly (WASM) library
2. C++ library compiled to WASM with Emscripten
   - Using the HDF5 C API (currently version 12.2.2)
   - Linking to pre-built https://github.com/usnistgov/libhdf5-wasm

- Source: https://github.com/usnistgov/h5wasm
- Package: https://www.npmjs.com/package/h5wasm

NIST CENTER FOR NEUTRON RESEARCH

Emscripten

- Compile C, C++ sources (or any language using LLVM) to webassembly
- Include Javascript loader *(optionally)*
  - With POSIX emulated filesystem (FS)
  - Additional FS features available, e.g.
    - IDBFS to persist FS across browser restart with builtin DB
    - NODERAWFS to use OS filesystem in nodejs
    - Lazy file loading: can read files as a stream
  - API for calling WASM functions, allocating memory (pointers)
  - Limited support for dynamic linking:
    - MAIN_MODULE and SIDE_MODULE
- Use convenient emcmake and emmake wrappers

NIST CENTER FOR NEUTRON RESEARCH

Who is using h5wasm?

- myHDF5: https://myhdf5.hdfgroup.org/
  - Viewing local files for users
  - Viewing remote files by URL
  - 2d and 1d plotting, table view, etc...
- PaNOSC h5web: https://h5web.panosc.eu/h5wasm
  - Like myHDF5, plus supports NeXus namespace
  - H5web for VSCode extension > 10,000 installs
- NIST Center for Neutron Research:
  - NeXus viewer https://ncnr.nist.gov/ncnrdata/view/nexus-hdf-viewer.html
  - Specialized viewers for SANS, reflectometry instruments

NIST CENTER FOR NEUTRON RESEARCH

Requested: compression plugins for h5wasm

- https://github.com/usnistgov/h5wasm/issues/51
  - Specifically: asking for ZStandard

- https://gitlab.esrf.fr/ui/myhdf5/-/issues/3

Support for filter plugins in h5wasm

- h5wasm is (now) built to allow dynamic linking!
  - as of v0.5.0, released 2023-05-15
  - using emscripten flag "MAIN_MODULE=2"
- Plugins can be used by
  - Compiling with emcc -s SIDE_MODULE=1
  - At runtime, write to emscripten virtual file system (FS)
    - Fetch file contents to Uint8Array
    - Write with FS.writeFile
    - Use destination path "/usr/local/hdf5/lib/plugin" (set at compile-time for h5wasm)

NIST CENTER FOR NEUTRON RESEARCH

Example: building ZStandard plugin

- Started with plugin sources from https://github.com/silx-kit/hdf5plugin/
  - Zstd_h5plugin.c
  - Zstd_h5plugin.h
  - CMakeLists.txt
- Make adjustments to build with Emscripten
  - Use pre-built libhdf5-wasm
  - Build dependencies as static libs

NIST CENTER FOR NEUTRON RESEARCH

ZStandard plugin: add HDF5 library

- In CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.14)
include(FetchContent)
FetchContent_Declare(
  libhdf5-wasm
  URL https://github.com/usnistgov/libhdf5-wasm/releases/download/v0.3.0_3.1.28/libhdf5-1_12_2-wasm.tar.gz
  URL_HASH SHA256=7089f9bf29dc3759d7aa77848cfa12d546eabd152d40dd00a90aace99c056600
)
FetchContent_MakeAvailable(libhdf5-wasm)
```

NIST CENTER FOR NEUTRON RESEARCH

ZStandard plugin: add libzstd

```
FetchContent_Declare(
   zstd
   GIT_REPOSITORY https://github.com/facebook/zstd
   SOURCE_SUBDIR build/cmake
   GIT_TAG v1.5.5
)
set(ZSTD_MULTITHREAD_SUPPORT OFF CACHE INTERNAL "")
set(ZSTD_BUILD_PROGRAMS OFF CACHE INTERNAL "")
set(ZSTD_BUILD_STATIC ON CACHE INTERNAL "")

FetchContent_MakeAvailable(zstd)
```

NIST CENTER FOR NEUTRON RESEARCH

ZStandard plugin: add target

```
set(PLUGIN_SOURCES zstd_h5plugin.c)

# HDF5 plugin as static library
add_library(zstd_h5_plugin STATIC ${PLUGIN_SOURCES})
target_include_directories(zstd_h5_plugin PRIVATE "${zstd_SOURCE_DIR}/lib")
set_target_properties(zstd_h5_plugin PROPERTIES
   OUTPUT_NAME H5Zzstd
   POSITION_INDEPENDENT_CODE ON
)

target_link_libraries(zstd_h5_plugin hdf5-wasm libzstd_static)
```

NIST CENTER FOR NEUTRON RESEARCH

ZStandard plugin: custom .so target

- Need custom CMake command to combine libraries with emcc the Emscripten way?

```
# create combined library (including libzstd)
set(OUTPUT_FILE libH5Zzstd.so CACHE INTERNAL "Output file name")
add_custom_target(zstd_h5_plugin_shared ALL
   COMMAND
   ${CMAKE_C_COMPILER} -s SIDE_MODULE=1 libH5Zzstd.a ${zstd_BINARY_DIR}/lib/libzstd.a -o   ${CMAKE_CURRENT_BINARY_DIR}/${OUTPUT_FILE}
   DEPENDS zstd_h5_plugin libzstd_static
)

set(PLUGIN_DIR ${CMAKE_CURRENT_SOURCE_DIR}/../dist CACHE PATH "")
install(PROGRAMS ${CMAKE_CURRENT_BINARY_DIR}/${OUTPUT_FILE} DESTINATION ${PLUGIN_DIR})
```

NIST CENTER FOR NEUTRON RESEARCH

ZStandard plugin: build

- In a shell:
  - emcmake cmake -S . -B build
  - cd build && emmake make install
- Result:
  - libzstd_static: build/_deps/zstd-build/lib/libzstd.a
  - zstd_h5_plugin: build/libH5Zzstd.a
  - zstd_h5_plugin_shared: ../dist/libH5Zzstd.so
- Testing: read a dataset made with **hdf5plugin.Zstd()
  - Success!

NIST CENTER FOR NEUTRON RESEARCH

Next example: LZ4 plugin

- Compiling LZ4 plugin worked but had runtime errors
- Could not find
  - From <arpa/inet.h>: htonl, htons, ntohl, ntohs
  - From HDF5: H5allocate_memory, H5free_memory
- Solution: added these to compile options of h5wasm
  - -s EXPORTED_FUNCTIONS=['_H5Fopen', '_H5Fclose', '_H5Fcreate', '_malloc', '_free', '_stderr', '_memset', '_memcpy', '_htonl', '_htons', '_ntohl', '_ntohs', '_H5allocate_memory', '_H5free_memory']
  - No appreciable increase in library size (still 3.3 MB)

NIST CENTER FOR NEUTRON RESEARCH

Dynamic linking with h5wasm: extra symbols

- Symbols needed by plugin may not be exported
- Solution 1: Export all symbols for maximum flexibility
  - Compiler setting: MAIN_MODULE=1
  - stdlib is included
  - h5wasm library is 9.6 MB
- Solution 2: Export extra symbols as needed
  - Compiler setting: MAIN_MODULE=2
  - h5wasm library is 3.3 MB
  - Specify extra symbols in EXPORTED_FUNCTIONS
  - New plugins might require new release of h5wasm

NIST CENTER FOR NEUTRON RESEARCH

Other issues: Chrome max. plugin size

- ~~Chrome will not load plugins > 4kB in the main thread~~
  - ~~Can load in a web worker or preload~~
- Chrome increased the limit while I was working on this
  - Upper limit is now 8 MB
- Some plugins so far:
  - LibH5Zztd.so is 755 kB
  - libH5Zlz4.so is 860 kB
- Firefox and Safari don't seem to have this limit

NIST CENTER FOR NEUTRON RESEARCH

Make h5wasm plugins widely available

- Public plugin repo based on https://github.com/HDFGroup/hdf5_plugins
- Similar to h5py plugins at https://github.com/silx-kit/hdf5plugin
- Need contributing developers with knowledge of:
  - Emscripten
  - CMake for build specifications
  - Github actions for automated builds / testing
  - Specific HDF5 plugins and supporting libraries
- Deploy built plugins to public CDN
  - npmjs.com
  - Github releases

NIST CENTER FOR NEUTRON RESEARCH

Initial effort: h5wasm-plugins

- https://github.com/bmaranville/h5wasm-plugins
- Plugins compiled so far:
  - ZStandard
  - LZ4
  - BZip2
- Still to do:
  - Packaging for use in bundlers
  - Publish to npm
  - Add tests/testing infrastructure
  - Add custom plugin path
  - Add "install" command to place plugins in path

NIST CENTER FOR NEUTRON RESEARCH

Make h5wasm + plugins sustainable

- Currently:
  - Public repo in a private Github organization (usnistgov)
    - Pull requests must be accepted by organization member
    - Single lead contributor
  - No succession plan for when contributor retires someday
- Future:
  - Move to community-supported organization?
    - h5wasm repo (https://github.com/usnistgov/h5wasm)
    - libhdf5-wasm repo (https://github.com/usnistgov/libhdf5-wasm)
    - h5wasm-plugins repo (https://github.com/bmaranville/h5wasm-plugins) (NEW!)
  - Multiple members of organization
  - Needs appropriate license

NIST CENTER FOR NEUTRON RESEARCH

Conclusions

- Plugin infrastructure seems to work on the web
  - With recent changes in Chrome, no need to use special workers
- Adding new plugins will take dedicated effort
  - Compiling all dependencies from source
  - Adjusting build settings for emscripten
  - Troubleshooting in non-standard enviroment (browser developer console!)
  - Choose between exporting all symbols (bigger) and targeted export
- Deploying and maintaining plugins package
  - Should be shared community effort for sustainability

NIST CENTER FOR NEUTRON RESEARCH

Thanks to…

- HDF5 Group
- Aaron Lun (@LTLA)
  - packaging libhdf5-wasm for CMake
- Loïc Huder (@loichuder), Axel Bocciarelli (@axelboc)
  - Creating and maintaining h5web (silx-kit)
  - Adding features to h5wasm to support integration with h5web
  - TypeScript types for h5web
- silx-kit team
  - Plugin implementations for h5py (used to start h5wasm plugins)

NIST CENTER FOR NEUTRON RESEARCH

Abstract

H5wasm is a webassembly-based library for reading and writing HDF5 files, which can be used natively in a web browser or in a local nodejs environment. The library has no external runtime dependencies, and is used in some online HDF5 viewers that don't require server-side processing: https://h5web.panosc.eu/h5wasm and https://myhdf5.hdfgroup.org/

The community has requested more compression plugins (e.g. ZSTANDARD) for h5wasm beyond the (included) DEFLATE, SHUFFLE, FLETCHER32 and SCALEOFFSET filters. In my talk I will discuss issues associated with adding plugins to h5wasm

- For collaborative work on h5wasm, a change from single-maintainer in a private organization (github/usnistgov)
- Incomplete support for dynamic linking in emscripten (MAIN_MODULE/SIDE_MODULE)
- Complex dependency chains for some plugins (all libraries have to be compiled to WASM)
- Browser limitations (e.g. max 4KB dynamic WASM loading in Chrome)
- I will demonstrate a proof-of-concept build of h5wasm including a ZSTANDARD plugin, and discuss why I was not able to easily build an LZ4 plugin.
- We can discuss a shared effort on building a repository for h5wasm like the the h5py plugins at https://github.com/silx-kit/hdf5plugin, also based on https://github.com/HDFGroup/hdf5_plugins. We could use people with skills in CMake, Emscripten, TypeScript and of course the HDF5 C API.

NIST CENTER FOR NEUTRON RESEARCH

Using h5wasm with plugin:

```
import h5wasm from "h5wasm";
await h5wasm.ready; // Emscripten wasm loader…

const file_buffer  = await (await fetch("https://my.repo/data.h5")).arrayBuffer();
// write file to Emscripten filesystem:
h5wasm.FS.writeFile("data.h5",  new Uint8Array(file_buffer));
const plugin_buffer  = await (await fetch("https://my.repo/plugins/libH5Zlz4.so")).arrayBuffer();
// write plugin to filesystem:
h5wasm.FS.mkdirTree("/usr/local/hdf5/lib/plugin");
h5wasm.FS.writeFile("/usr/local/hdf5/lib/plugin/libH5Zlz4.so",  new Uint8Array(plugin_buffer));

const f = new h5wasm.File("data.h5",  "r");
f.get("data").value;  // decompressed  numbers
```