Moving HDF5 Applications from One Major Release to Another

July 31, 2019

The HDF Group

Elena Pourmal

epourmal@hdfgroup.org

HDF5 Mission

The HDF Group Mission

- To provide the highest quality software for managing large complex data sets,

- To provide outstanding services for users of HDF technologies, and

- To insure long-term access and usability of data that is stored using HDF technologies.

HDF5 Software

- Free to everyone
- New features, security patches, bug fixes are available to all users
  - HDF5 source code and binary releases
    - Found on HDF portal
      https://portal.hdfgroup.org/display/support/Downloads
    - Scheduled for May and November of each year for maintenance releases
    - New major releases are announced when we are ready to start a new series of a maintenance release
    - We ask you to login to our portal **only when you want to download binaries**
  - HDF5 source in Bitbucket
    https://bitbucket.hdfgroup.org/projects/HDFFV

Today’s Goal

- **Help you to decide when and how to move to new major HDF5 release**
  - Navigating through HDF5 releases
  - Provide guidance on moving applications to a new HDF5 release
  - Present HDF5 Roadmap for 2019 – 2020

Outline

- **Motivation**
- **Navigating through HDF5 Versioning**
- **HDF5 backward and forward compatibility**
- **Migration Guidance**
  - How to move an application from one release to another?
  - What to consider?
- **HDF5 Roadmap and release schedule**
- **Future events**

Motivation

- **Why do we create new versions of the library and extensions to the HDF5 file format?**
- **HDF5 is still under active development**
  - Bug fixes and security patches
  - Performance improvements
  - New features
  - All of the above may require *file format changes*, API changes or new APIs, public structure changes, etc.
- **The HDF Group is committed to minimum disruption to your development process**

HDF5 Backward/Forward Compatibility

- **What do we promise to our users?**
  - File Format
    - *A newer version of the library will always read files created with an older version*
    - *An application that doesn’t use new features of the newer library will create files compatible with the older version of the library*
  - Library APIs
    - *An application written for the older library will always compile and link with the newer library*

HDF5 Backward/Forward Compatibility (cont'd)

- **The next slides explain in more detail how we achieve HDF5 backward/forward compatibility**
- **We will talk about factors you need to be aware before you make a decision to move to new HDF5 release**
  - release versioning
  - development and release process
  - file format versioning
  - API versioning

Navigating through HDF5 Versioning

Releases, File Format, APIs and backward/forward compatibility

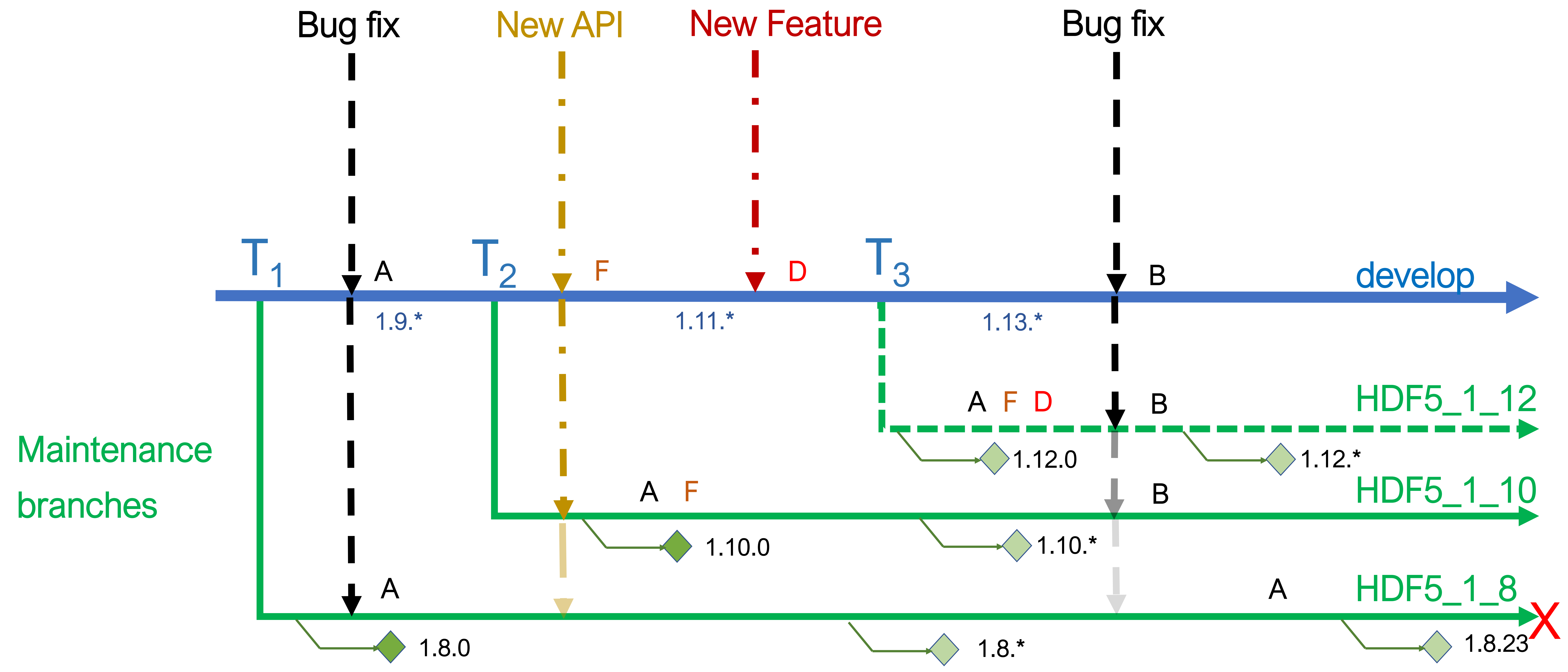
HDF5 Release Versioning

- **See the HDF5 Library Release Version Numbers document**
- **HDF5 version number has a form X.Y.Z(-suffix)**
  - X is the major version number
  - Y is the minor version number
  - Z is the release number
  - The suffix is present in software snapshots and release candidates
  - Examples of how we refer to our releases:
    - 1.10.**0 major** release
    - 1.10.**5 maintenance** release
    - 1.10.5-snap0 snapshot
      - First software snapshot after 1.10.5 release
    - 1.10.6-pre2 release candidate
      - Second pre-release candidate of the 1.10.6 maintenance release

What to expect when Y or Z changes?

- **X didn’t change in 21 years**
  - We are contemplating dropping X from versioning
- **Y is even and increased by 2 when any of the following changes is present in the new release:**
  - File Format extensions to accommodate new features
  - Changes in public data structures
  - Changes in APIs default signatures
  - Architectural changes in the library
  - ABI non-compatible with X.Y-2.* releases
- **Z is increased by 1 when a new version of HDF5 1.Y is available**
  - Includes:
    - Bug fixes
    - New features and APIs
    - Performance enhancements
  - ABI compatible with X.Y.0 to X.Y.Z-1 releases

HDF5 Development Process



HDF5 File Format Versioning

- **There is <u>no</u> HDF5 file format version number**
  - Micro-versioning: each object and structure within an HDF5 file is versioned
  - An updated “File Format Specification” is available with every major release
  - There is no way to find what version of the HDF5 library created or modified a particular file
  - Features specified by an application trigger specific object micro-versioning; see the “Setting Bounds for Object Creation in HDF5 1.10.0” document for more information

HDF5 File Format Forward Compatibility

- **We adhere to the maximum file format compatibility principle**
  - By default, HDF5 files are written with the earliest version of the object that describes information, rather than always using the latest possible
    - Assures best forward compatibility with the older versions of the library since objects in the files created by the newer version of the library can be accessed by the earlier version of the library
  - With the HDF5 1.10.2 release we provide a mechanism to control objects' versions based on major releases 1.8, 1.10, and 1.12, so older versions can read the file created by the newer version of the library
    - E.g., HDF5 1.10.* will create objects with the versions known to HDF5 1.8.0 or earlier

HDF5 API Versioning

- **In the HDF5 1.8.0 release and later some functions have a version suffix**
- **Example:**
  - HDF5 1.6.*

    H5Gcreate (loc_id, "New/My old group", 0)
  - HDF5 1.8.0 and later

    *H5Gcreate**1** (loc_id, "New/My new group", 0)*

    *H5Gcreate**2** (loc_id, "New/My new group", lcpl_id, gcpl_id, gapl_id)*
  - New features can be invoked when using the latest version of the function
    - Creation order
    - Unicode names
    - Compact storage
    - Intermediate group creation

HDF5 API Compatibility Macros

- **A function without a suffix is a macro. Example:**

  H5Gcreate is a macro to

  *H5Gcreate**1*** (*loc_id, "New/My new group", 0*)

  *H5Gcreate**2*** (*loc_id, "New/My new group", lcpl_id, gcpl_id, gapl_id*)

- **By default the macro maps to the latest version of the function**
  - Can be overwritten with compiler flags, configuration flags, etc.
- **See more about API compatibility macros on the HDF Portal**
  https://portal.hdfgroup.org/display/HDF5/API+Compatibility+Macros

Migration Guidance

How and when to move applications? What to expect?

Moving to a new HDF5 Release

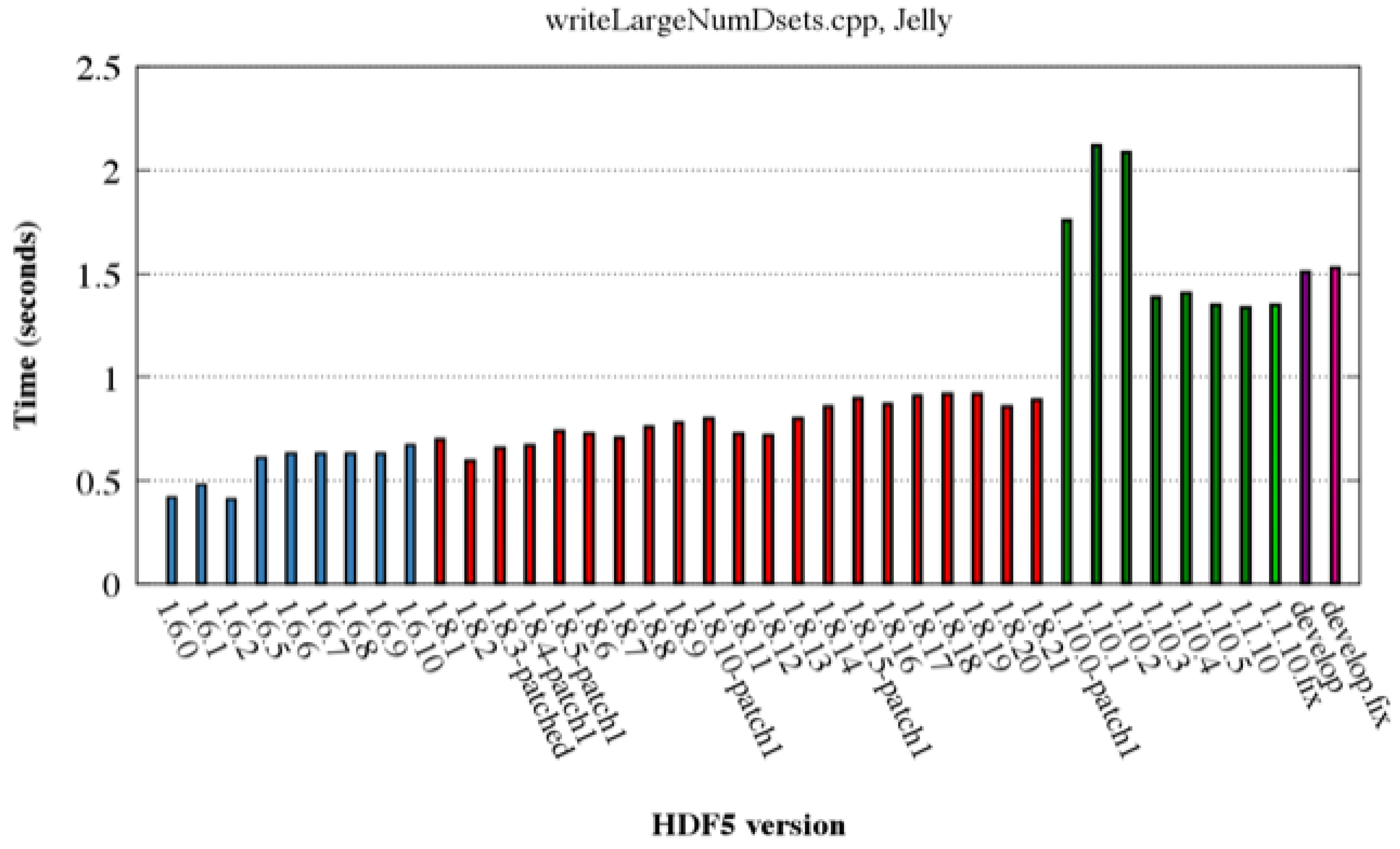
- **We always encourage our users to move to the latest release**
  - See https://portal.hdfgroup.org
  - You can find information about the releases including new features under the Release Specific Information page
- **If you are using previous HDF5 1.10 releases you should be able to relink with the new version of HDF5 1.10 without recompiling application**
  - Maintenance releases are ABI (application binary interface) compatible
    - See ABI reports and ABI compatibility issues in the Software changes from release to release for HDF5 1.10 document.

Moving to new HDF5 Release (cont'd)

- **If you are using HDF5 1.8.* or earlier you will need to recompile your application**
  - Major releases are not ABI compatible
- **Issues one may encounter:**
  - A performance drop in current HDF5 1.10.* releases compared to HDF5 1.8 and earlier releases
  - An application will not compile due to an API version change
  - An application may create files that cannot be read by the earliest versions of the library (file format forward compatibility).

Performance Issue

- **Known performance problem with the latest 1.10**
  - HDF5 1.8.* and earlier perform better for some use cases
  - Check the performance of your application and report the problem to help@hdfgroup.org or HDF-FORUM
  - We are working on solutions



Compilation failure: Using old APIs with the newer releases

- **We encourage applications to use the latest versions of APIs**
- **Options**
  - When using pre-built HDF5 binaries
    - Use the compiler option to specify the API version; works well for dealing with multiple API changes
      - Example: -DH5_USE_16_API or –DH5_USE_18_API
    - Use the compiler option to indicate a specific API
      - Example: -DH5Rdereference_vers=1
  - When building your own HDF5 binary
    - Use the configure flag to specify the API version
      - Example: --with-default-api-version=v18
  - Check the libhdf5.settings file found in the lib subdirectory of the installation directory
  - Check "API Compatibility Macros" document for more information about a specific APIs versions

Breaking File Format Forward Compatibility

- **File Format Forward Compatibility may break if an application uses:**
  - **H5Pset_libver_bounds** to trigger the latest features of HDF5 1.8

    Example: H5Pset_libver_bounds (*H5F_LIBVER_LATEST, H5F_LIBVER_LATEST)*
    - Change the application to use *H5F_LIBVER_v18* for the first argument

      Example: H5Pset_libver_bounds (*H5F_LIBVER_V18*, *H5F_LIBVER_LATEST)*
  - **New features**, for example, in HDF5 1.10.*
    - Virtual Dataset (VDS)
    - Single Writer /Multiple Reader
    - Paged allocation, etc.
    - See the Software changes from release to release for HDF5 1.10 document.

Making HDF5 1.10 File Compatible with 1.8

- **One can use the HDF5 tools to make a file created by the HDF5 1.10 libraries compatible with earlier versions of HDF5**
  - **h5repack**
    - Provides several conversion options flags
      - "-l" to change layout (VDS)
      - "-high=H5FLIBVER_V18"
    - Requires all data be rewritten
  - **h5format_convert**
    - "Downgrades" the file to a 1.8 compatible file format without rewriting raw data
    - Doesn't affect VDS
  - **h5clear**
    - Repairs a file written in SWMR mode
    - Removes cache image

Demo

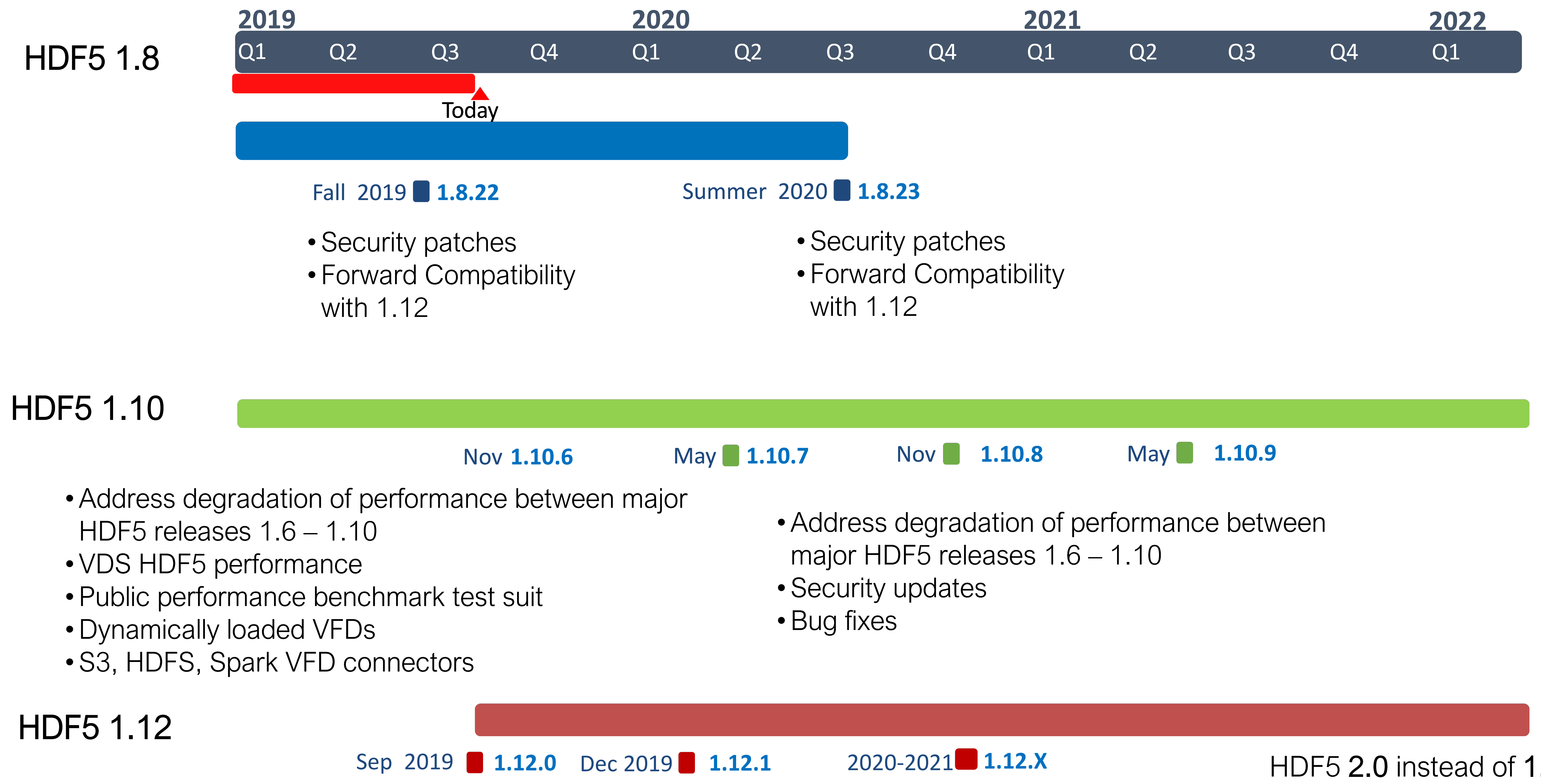
- **Compile and run a 1.6 application to create a file**
  - Use the 1.10 version of h5dump to display the file (backward compatibility)
- **Compile and run a 1.6 application with the 1.10 library**
  - Show how to deal with API compatibility
  - Use the 1.6 and 1.8 version of h5dump to display the file (forward compatibility)
- **Modify a 1.6 application to use the latest format**
  - Use the 1.8 version of h5dump to show broken forward compatibility
  - Modify an application to create 1.8 compatible file
- **Modify an application to use SWMR**
  - Use the 1.8 version of h5dump to demonstrate broken forward compatibility
  - Use h5format_convert and h5repack and dump the data with the 1.8 version of h5dump

HDF5 Roadmap for 2019 - 2020

What is coming?

HDF5 Roadmap 2019 - 2020

The HDF Group



HDF5 1.8

Fall 2019 1.8.22

- Security patches
- Forward Compatibility with 1.12

Summer 2020 1.8.23

- Security patches
- Forward Compatibility with 1.12

HDF5 1.10

- Address degradation of performance between major HDF5 releases 1.6 – 1.10
- VDS HDF5 performance
- Public performance benchmark test suit
- Dynamically loaded VFDs
- S3, HDFS, Spark VFD connectors

- Address degradation of performance between major HDF5 releases 1.6 – 1.10
- Security updates
- Bug fixes

HDF5 1.12

- VOL architecture
- References
- Parallel compression (enhanced)

- VOL plugins (DAOS)
- VFD plugins
- Dynamically loaded VFDs
- S3, HDFS, SPARK VFD connectors

- Query and Indexing
- Full SWMR
- Mirror VFD
- Provenance (Onion) VFD

HDF5 2.0 instead of 1.12.0?
Storage beyond the current File Systems and HDF5 file format

Upcoming Events

Want to learn more?

Future Events

- September 11, 2019
  - HDF5 1.12 Features via WebEx
    - Link to register on the Forum and in chat
- September 17 – 18, 2019
  - HDF5 European Workshop for Science and Industry, Grenoble, France
    - Free
    - Followed by h5py code camp on Sept 19 and 20
    - Visit hdfgroup.org/conferences for more info
- October 6, 2019
  - HDF5 Workshop at ICALEPCS 2019
- November 17 – 22, 2019
  - HDF5 BoF (proposed) @ SC19

Thank you!

Questions?