xnetwork

Johan Mabille and Sylvain Corlay

Jul 13, 2021

INSTALLATION

Basic tools (containers, algorithms) used by other quantstack packages

xnetwork gathers generic purpose algorithms and containers that are used by the *xtensor* stack and the *xeus* stack.

Some of the features are C++14 backport of C++17 classes and algorithms, such as *variant* or *any*.

CHAPTER

ONE

LICENSING

We use a shared copyright model that enables all contributors to maintain the copyright on their contributions.

This software is licensed under the BSD-3-Clause license. See the LICENSE file for details.

1.1 Installation

Although `xnetwork` is a header-only library, we provide standardized means to install it, with package managers or with cmake.

Besides the xnetwork headers, all these methods place the `cmake` project configuration file in the right location so that third-party projects can use cmake's `find_package` to locate xnetwork headers.

1.1.1 Using the conda-forge package

A package for xnetwork is available for the mamba (or conda) package manager.

```
mamba install -c conda-forge xnetwork
```

1.1.2 Using the Spack package

A package for xnetwork is available on the Spack package manager.

```
spack install xnetwork
spack load xnetwork
```

1.1.3 From source with cmake

You can also install `xnetwork` from source with cmake. On Unix platforms, from the source directory:

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/path/to/prefix ..
make install
```

On Windows platforms, from the source directory:

```
mkdir build
cd build
cmake -G "NMake Makefiles" -DCMAKE_INSTALL_PREFIX=/path/to/prefix ..
nmake
nmake install
```

See the section of the documentation on *Build and configuration*, for more details on how to cmake options.

1.2 Changelog

1.2.1 0.7.2

- Removed C++14 code from xbase_fixed_string

1.2.2 0.7.1

- test/test_xsystem.cpp: Allow for ctest executable names
- Fixed return type in xbasic_fixed_string to be compatible with C++11

1.2.3 0.7.0

- Added missing FreeBSD headers
- Fixed call to sysctl
- Added meta switch
- Implement partial dispatch
- Improve xbasic fixed string

1.2.4 0.6.23

- Added `xnetwork::executable_path` and `xnetwork::prefix_path`

1.2.5 0.6.22

- Handle complex promote type w/ different T1 and T2
- Install as arch dependent for cmake > 3.14

1.2.6 0.6.21

- CMake: Modernized GTest integration
- Add support for `xnetwork::endianness`

1.2.7 0.6.20

- Added xtraits for future simple specialization
- Added half_float implementation

1.2.8 0.6.19

- Improved `mpl::contains` implementation
- Added `are_equivalent_sequences`
- Various minor improvements

1.2.9 0.6.18

- Relaxed dimension constraint on multidispatcher
- Replaced throw with XNETWORK_THROW to support disabling exceptions

1.2.10 0.6.17

- Implemented `index_of` for `mpl::vector`
- Implemented multimethods pattern
- Implemented visitor pattern

1.2.11 0.6.16

- Fixed mpark variant inclusion guards
- Add a serialiser for xvariant to json
- Removed capture all by reference

1.2.12 0.6.15

- Renamed mpark variant header inclusion guard

1.2.13 0.6.14

- Implemented value iterator for map containers

1.2.14 0.6.13

- mpark/variant small change for CUDA 10.2 workaround
- Switched the documentation build to QuantStack channel
- Refactored CI

1.2.15 0.6.12

- NVCC CUDA compiler compatibility
- Wrapped call to `find_package`

1.2.16 0.6.11

- Avoids C++20 “requires” keyword

1.2.17 0.6.10

- Set up xnetwork target’s public headers
- CMake: adding C++14 standard to target

1.2.18 0.6.9

- Implemented stepping iterators

1.2.19 0.6.8

- Fixed murmur implementation for x86 platform

1.2.20 0.6.7

- Specialized `promote_type` for `std::complex`

1.2.21 0.6.6

- Fixed `promote_type` for `std::chrono::time_point`
- Update README for Conan installation instructions

1.2.22 0.6.5

- Add supports for clang-cl compiler
- Fix cmake command
- Fix compiler error with clang-cl compiler

1.2.23 0.6.4

- Fixed forward type

1.2.24 0.6.3

- Fix constness issue in xnetwork's implementation of std::any.

1.2.25 0.6.2

- Allows xnetwork to build with -fno-exceptions
- Added `constify` and `constify_t`
- Added `size_t` overloads for random access iterators

1.2.26 0.6.1

- Latex does not know how to include svg
- Added `XNETWORK_REQUIRES_IMPL` macro
- Removed warnings

1.2.27 0.6.0

- Standalone build of xnetwork tests
- Moved `xmasked_value` from *xtensor*
- Moved `promote_type` from *xtensor*
- Disabled `xoptional` methods for `xmasked_value`
- Implemented `select`
- `make_sequence` from `initializer_list`

1.2.28 0.5.4

- Implementation of mpl::unique
- Prevent installation of gtest artifact

1.2.29 0.5.3

- upgraded to mpark/variant 1.4.0
- implemented concepts
- implemented split of type lists

1.2.30 0.5.2

- fixed C++11 compatibility in xjson.hpp

1.2.31 0.5.1

- reverse order of initialization of optional
- fixup mime type rendering of fixed string
- closure wrapper assignment fixed

1.2.32 0.5.0

- Serialization and deserialization of fixed strings
- Inequality comparisons removed from bidirectional iterator base
- Simplified forward sequence
- Fixed forward sequence
- Removed warnings
- const reference getter for variant holding non const references
- xget on rvalue fixed
- Added storage option to fixed string
- Added missing entries of header files in CMakeLists.txt

- Refactored xdynamic_bitset
- Fixed forwrad sequence for non resizable types
- Removed meta pop-back

1.2.33 0.4.16

- meta find_if implementation
- Enable CTest and CMake cleanup
- Make nlohmann_json optional in the tests, exported C++14 requirements

1.2.34 0.4.15

- Value types in const closures are not const qualified anymore, to allow move
- Added third template parameter to forward_sequence that allows for true forwarding of sequences

1.2.35 0.4.14

- Fixed typo in 'xnetwork.pc.in'
- Removed -march=native from systems that do not support in CMakeLists
- Added hash.verification result for big-endian systemss
- Fixed common_optional_impl
- Implemented xeus-cling mime_bundle_repr for xoptional, xcomplex and xfixed_string

1.2.36 0.4.13

- CMake call to find_package with nlohmann_json is QUIET
- Fix typo in xoptional swap
- Added pkgconfig support

1.2.37 0.4.12

- operator overload fixes for xcomplex

1.2.38 0.4.11

- add missing <*limits*> header in xcomplex
- fix xcomplex isnan test

1.2.39 0.4.10

- *xcomplex* implementation
- *xcomplex_sequence* implementation

1.2.40 0.4.9

- return type of *static_if* fixed

1.2.41 0.4.8

- support for JSON serialization of xoptionals

1.2.42 0.4.7

- support for uninitialized *make_sequence*

1.2.43 0.4.6

- remove an unused file.
- support for overloaded lambdas

1.2.44 0.4.5

- xget for variant on xclosure_wrapper

1.2.45 0.4.4

- bug fix in any
- hierarchy generators

1.2.46 0.4.3

- missing near integers functions for *xoptional*
- *xoptional* compilation issue fixed

1.2.47 0.4.2

- added missing operators for xoptional
- removed compiler warning if cpp_exceptions already defined

1.2.48 0.4.1

- Bug fix in move semantics for xoptional free functions (*value* and *has_value*)
- Use *static_if* instead of regular *if* to remove gcc-6 warning.
- Document installation with the Spack package manager.
- Fix complex operators with closure wrappers.
- Integrate upstream fix for the variant implementation.

1.2.49 0.4.0

- Migration to modern target-based cmake

1.2.50 0.3.9

- Bug fix in the computing of hashes for 32 bit platforms
- Fixing warnings

1.2.51 0.3.8

- Improvements and fixes in base iterators (common iterator tag)

1.2.52 0.3.7

- Fixes in *xoptional*.

1.2.53 0.3.6

- Addition of base iterators for linear containers, and associative containers.

1.2.54 0.3.5

- Addition of *value* and *has_value* free functions.
- Bug fix in comparison operator for *xclosure_wrapper*.

1.2.55 0.3.4

- Better semantics for assignment operators in *xoptional*.
- Addition of *static_if* in *xnetwork::mpl*.
- Addition of *xnetwork::identity* functor in xfunctional.

1.2.56 0.3.3

- Work around Visual Studio compiler bug in *xoptional_proxy*.

1.2.57 0.3.2

- Improvement of xoptional value semantics (explicit constructors when underlying value type not implicitely constructable)

1.2.58 0.3.1

- Fixes in closure wrapper semantics

1.2.59 0.3.0

- Improve optional sequence
- Use dynamic bitset in optional vector
- Added base64encode and base64decode

1.2.60 0.2.11

- Added dynamic bitset

1.2.61 0.2.10

- Added meta programming tools

1.2.62 0.2.9

- Added variant implementation

1.2.63 0.2.8

- Added proxy wrapper for pointer semantics.

1.2.64 0.2.7

- Added implementation for closure pointer

1.2.65 0.2.6

- Added base class for random access iterators

1.2.66 0.2.5

- Added closure wrappers

1.2.67 0.2.4

- Added implementation of std::any

1.2.68 0.2.3

- Fixed bug in fixed-size string hashing

1.2.69 0.2.2

- Added the hashing of fixed-size strings

1.2.70 0.2.1

- Fixed-size strings
- Fixup issue with ambiguous overload of operator<<

1.2.71 0.2.0

- Moving features from xtensor (xcomplex, xoptional, xsequence, xtypetraits)

1.3 Basic types

1.3.1 any

xnetwork::any is a backport of the C++17 class *std::any*. The class describes a type-safe container for single values of any type:

```
#include <iostream>
#include "xnetwork/xany.hpp"

xnetwork::any a = 1;
std::cout << a.type().name() << ": " << xnetwork::any_cast<int>(a) << std::endl;
// => i: 1

try
{
    std::cout << xnetwork::any_cast<float>(a) << std::endl;
}
catch(const xnetwork::bad_any_cast& e)
{
    std::cout << e.what() << std::endl;
}
// => bad any_cast

a.reset();
std::cout << std::boolalpha << a.empty() << std::endl;
// => true
```

The API of *xnetwork::any* is the same as that of *std::any*. Full documentation can be found on cppreference.

1.3.2 xbasic_fixed_string

TODO

1.3.3 xcomplex

xcomplex is the equivalent to *std::complex*, where the real ang imaginary part can be stored either as values, or as references. Therefore, it can be used as a proxy on real values already initialized. This is particularly interesting for storing real and imaginary parts in different containers, and gather them as complex values for computation. This allows optimzations (such as vectorization) on the real and imgaginary values.

```
#include <iostream>
#include <vector>
#include "xnetwork/xcomplex.hpp"

std::vector<double> arg1_real = { 1., 2.};
std::vector<double> arg1_imag = { 3., 4.};
std::vector<double> arg2_real = { 2., 4.};
std::vector<double> arg2_real = { 1., 3.};
std::vector<double> res_real(2);
std::vector<double> res_imag(2);

using complex = xnetwork::xcomplex<double&, double&>;
using const_complex = xnetwork::xcomplex<const double&, const double&>;
```

(continues on next page)

(continued from previous page)

```
for (size_t i = 0; i < 2; ++i)
{
    complex res(res_real[i], res_img[i]);
    res = const_complex(arg1_real, arg1_imag) * const_complex(arg2_real, arg2_imag);
    std::cout << "res = (" << res.real(), << ", " << res.imag() << std::endl;
}
```

The API of *xnetwork::xcomplex* is the same as that of *std::complex*, with the ability to store values as references. Full documentation can be found on cppreference.

1.3.4 half_float

The *half_float* class implements an IEEE-conformant half-precision floating-point type with the usual arithmetic operators and conversions. It is implicitly convertible from single-precision floating-point, which makes expressions and functions with mixed-type operands to be of the most precise operand type.

```
#include <iostream<
#include "xnetwork/xhalf.hpp"

xnetwork::half_float f0 = 1.0f;
xnetwork::half_float f1 = 2.0f;
auto res = f0 + f1;
std::cout << res << std::endl;
```

1.3.5 xmasked_value

TODO

1.3.6 xoptional

TODO

1.3.7 xvariant

TODO

1.4 Containers

1.4.1 xcomplex_sequence

TODO

1.4.2 xdynamic_bitset

TODO

1.4.3 xiterator_base

TODO

1.4.4 xoptional_sequence

TODO

1.4.5 xsequence

TODO

1.4.6 xspan

TODO

1.5 Meta Programming

1.5.1 xclosure

TODO

1.5.2 xtype_traits

TODO

1.5.3 type vector

TODO

1.6 Design Patterns

1.6.1 Hierarchy Generator

TODO

1.6.2 Multimethods

TODO

1.6.3 Visitor

TODO

1.7 Miscellaneous

1.7.1 Base 64 encoding

TODO

1.7.2 xhash

TODO

1.7.3 endianness

TODO

1.8 Build and configuration

1.8.1 Build

`xnetwork` build supports the following options:

- `BUILD_TESTS`: enables the `xtest` target (see below).
- `DOWNLOAD_GTEST`: downloads `gtest` and builds it locally instead of using a binary installation.
- `GTEST_SRC_DIR`: indicates where to find the `gtest` sources instead of downloading them.
- `XNETWORK_DISABLE_EXCEPTIONS`: indicates that tests should be run with exceptions disabled.

All these options are disabled by default. Enabling `DOWNLOAD_GTEST` or setting `GTEST_SRC_DIR` enables `BUILD_TESTS`.

If the `BUILD_TESTS` option is enabled, the following target is available:

- xtest: builds an run the test suite.

For instance, building the test suite of `xnetwork` where the sources of `gtest` are located in e.g. `/usr/share/gtest`:

```
mkdir build
cd build
cmake -DGTEST_SRC_DIR=/usr/share/gtest ../
make xtest
```

1.9 Releasing xnetwork

1.9.1 Releasing a new version

From the relevant branch of xnetwork

- Make sure that you are in sync with the master branch of the upstream remote.
- In file `xnetwork_config.hpp`, set the macros for `XNETWORK_VERSION_MAJOR`, `XNETWORK_VERSION_MINOR` and `XNETWORK_VERSION_PATCH` to the desired values.
- In file `README.md`, modify the binder link to point to the new release.
- Stage the changes (`git add`), commit the changes (`git commit`) and add a tag of the form `Major.minor.patch`. It is important to not add any other content to the tag name.
- Push the new commit and tag to the main repository. (`git push`, and `git push --tags`)

1.9.2 Updating the conda-forge recipe

xnetwork has been packaged for the conda package manager. Once the new tag has been pushed on GitHub, edit the conda-forge recipe for xnetwork in the following fashion:

- Update the version number to the new `Major.minor.patch`.
- Set the build number to `0`.
- Update the hash of the source tarball.
- Check for the versions of the dependencies.
- Optionally, rerender the conda-forge feedstock.