xtensor-zarr

Wolf Vollprecht, Johan Mabille and Sylvain Corlay

Dec 01, 2021

INSTALLATION

1 Enabling xtensor-zarr in your C++ libraries **3**

2 Licensing **5**
- 2.1 Installation . . . 5
- 2.2 Basic Usage . . . 6
- 2.3 API Reference . . . 9
- 2.4 Build and configuration . . . 11
- 2.5 Releasing xtensor-zarr . . . 12

Index **13**

Implementation of the Zarr version 3.0 core protocol based on the xtensor C++ multi-dimensional array library.

xtensor-zarr implements Zarr (v3) for reading and writing chunked arrays. It also implements the store interface, compressors (e.g. GZip, Blosc), and IO handlers (e.g. local file system, Google Cloud Storage).

CHAPTER

ONE

ENABLING XTENSOR-ZARR IN YOUR C++ LIBRARIES

`xtensor`, `xtensor-io` and `xtensor-zarr` require a modern C++ compiler supporting C++14. The following C++ compilers are supported:

- On Windows platforms, Visual C++ 2015 Update 2, or more recent
- On Unix platforms, gcc 4.9 or a recent version of Clang

CHAPTER

TWO

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.

2.1 Installation

xtensor-zarr is a header-only library but depends on some traditional libraries that need to be installed. On Linux, installation of the dependencies can be done through the package manager, anaconda or manual compilation.

2.1.1 Using the conda package

A package for xtensor-zarr is available on the conda package manager. The package will also pull all the dependencies (xtensor, xtensor-io).

```
conda install xtensor-zarr -c conda-forge
```

The easiest way to make use of xtensor-zarr in your code is by using cmake for your project. In order for cmake to pick up the xtensor-zarr dependency, just utilize the interface target and link the xtensor-zarr library to your target.

```
add_executable(my_exec my_exec.cpp)
target_link_libraries(my_exec
  PUBLIC
    xtensor-zarr
)
```

This should be enough to add the correct directories to the include_directories and link the required libraries. However, depending on your system setup there might be problems upon executing, as the dynamic library is not picked up correctly. So if you run into errors that read something like "Library could not be opened ... ", then set the `RPATH`, the runtime library search path, to the `conda` library path of your environment. We utilize the `CMAKE_INSTALL_PREFIX` for this purpose, so if you call cmake like this `cmake .. -DCMAKE_INSTALL_PREFIX=$CONDA_PREFIX`, add the following to your `CMakeLists.txt`.

```
set_target_properties(my_exec
  PROPERTIES
    INSTALL_RPATH "${CMAKE_INSTALL_PREFIX}/lib;${CMAKE_INSTALL_PREFIX}/${CMAKE_INSTALL_
↪LIBDIR}"
    BUILD_WITH_INSTALL_RPATH ON
)
```

2.1.2 From source with cmake

You can also install `xtensor-zarr` from source with cmake. However, you need to make sure to have the required libraries available on your machine.

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
```

2.2 Basic Usage

2.2.1 Create a hierarchy

```
#include "xtensor-zarr/xzarr_hierarchy.hpp"
#include "xtensor-zarr/xzarr_file_system_store.hpp"

int main ()
{
    // create a hierarchy on the local file system
    xt::xzarr_file_system_store store("test.zr3");
    auto h = xt::create_zarr_hierarchy(store);
}
```

2.2.2 Open a hierarchy

```
#include "xtensor-zarr/xzarr_hierarchy.hpp"
#include "xtensor-zarr/xzarr_file_system_store.hpp"

int main ()
{
    // open a hierarchy
    xt::xzarr_file_system_store store("test.zr3");
    auto h = xt::get_zarr_hierarchy(store);
}
```

2.2.3 Create an array

```cpp
#include <vector>
#include "xtensor-zarr/xzarr_hierarchy.hpp"
#include "xtensor-zarr/xzarr_file_system_store.hpp"
#include "xtensor-io/xio_binary.hpp"

int main ()
{
    // open a hierarchy
    xt::xzarr_file_system_store store("test.zr3");
    auto h = xt::get_zarr_hierarchy(store);
    // create an array in the hierarchy
    nlohmann::json attrs = {{"question", "life"}, {"answer", 42}};
    using S = std::vector<std::size_t>;
    S shape = {4, 4};
    S chunk_shape = {2, 2};
    xt::zarray a = h.create_array(
        "/arthur/dent",  // path to the array in the store
        shape,  // array shape
        chunk_shape,  // chunk shape
        "<f8",  // data type, here little-endian 64-bit floating point
        'C',  // memory layout
        '/',  // chunk identifier separator
        xt::xio_binary_config(),  // compressor (here, raw binary)
        attrs,  // attributes
        10,  // chunk pool size
        0.  // fill value
    );
    // write array data
    a(2, 1) = 3.;
}
```

2.2.4 Access an array

```cpp
#include <iostream>
#include "xtensor-zarr/xzarr_hierarchy.hpp"
#include "xtensor-zarr/xzarr_file_system_store.hpp"

int main ()
{
    // open a hierarchy
    xt::xzarr_file_system_store store("test.zr3");
    auto h = xt::get_zarr_hierarchy(store);
    // access an array in the hierarchy
    xt::zarray a = h.get_array("/arthur/dent");
    // read array data
    std::cout << a(2, 1) << std::endl;
    // prints `3.`
    std::cout << a(2, 2) << std::endl;
    // prints `0.` (fill value)
```

(continues on next page)

(continued from previous page)

```
    std::cout << a.attrs() << std::endl;
    // prints `{"answer":42,"question":"life"}`
}
```

2.2.5 Create a group

```
#include "xtensor-zarr/xzarr_hierarchy.hpp"
#include "xtensor-zarr/xzarr_file_system_store.hpp"

int main ()
{
    xt::xzarr_file_system_store store("test.zr3");
    auto h = get_zarr_hierarchy(store);
    nlohmann::json attrs = {{"heart", "gold"}, {"improbability", "infinite"}};
    // create a group
    auto g = h.create_group("/tricia/mcmillan", attrs);
}
```

2.2.6 Explore the hierarchy

```
#include <iostream>
#include "xtensor-zarr/xzarr_hierarchy.hpp"
#include "xtensor-zarr/xzarr_file_system_store.hpp"

int main ()
{
    xt::xzarr_file_system_store store("test.zr3");
    auto h = get_zarr_hierarchy(store);
    // get children at a point in the hierarchy
    std::string children = h.get_children("/").dump();
    std::cout << children << std::endl;
    // prints `{"arthur":"implicit_group","marvin":"explicit_group","tricia":"implicit_
↪group"}`
    // view the whole hierarchy
    std::string nodes = h.get_nodes().dump();
    std::cout << nodes << std::endl;
    // prints `{"arthur":"implicit_group","arthur/dent":"array","tricia":"implicit_group
↪","tricia/mcmillan":"explicit_group"}`
}
```

2.2.7 Use cloud storage

```cpp
#include <iostream>
#include "xtensor-zarr/xzarr_gcs_store.hpp"

int main ()
{
    // create an anonymous Google Cloud Storage client
    gcs::Client client((gcs::ClientOptions(gcs::oauth2::CreateAnonymousCredentials())));
    xzarr_gcs_store s1("zarr-demo/v3/test.zr3", client);
    // list keys under prefix
    auto keys1 = s1.list_prefix("data/root/arthur/dent/");
    for (const auto& key: keys1)
    {
        std::cout << key << std::endl;
    }
    // prints:
    // data/root/arthur/dent/c0/0
    // data/root/arthur/dent/c0/1
    // data/root/arthur/dent/c1/0
    // data/root/arthur/dent/c1/1
    // data/root/arthur/dent/c2/0
    // data/root/arthur/dent/c2/1

    xzarr_gcs_store s2("zarr-demo/v3/test.zr3/meta/root/marvin", client);
    // list all keys
    auto keys2 = s2.list();
    for (const auto& key: keys2)
    {
        std::cout << key << std::endl;
    }
    // prints:
    // android.array.json
    // paranoid.group.json
}
```

2.3 API Reference

2.3.1 Hierarchy

Defined in `xtensor-zarr/xzarr_hierarchy.hpp`

template<class **store_type**>

class **xzarr_hierarchy**

Zarr hierarchy handler.

The *xzarr_hierarchy* class implements a handler for creating and accessing a hierarchy, an array or a group, as well as exploring the hierarchy.

See zarray, xzarr_group, xzarr_node

tparam store_type The type of the store (e.g. *xzarr_file_system_store*)

Warning: doxygenfunction: Unable to resolve function "xt::create_zarr_hierarchy" with arguments None in doxygen xml output for project "xtensor-zarr" from directory: ../xml. Potential matches:

```
- template<class store_type> xzarr_hierarchy<store_type> create_zarr_hierarchy(store_
↪type &store, const std::string &zarr_version = "3")
- xzarr_hierarchy<xzarr_file_system_store> create_zarr_hierarchy(const char *local_
↪store_path, const std::string &zarr_version = "3")
- xzarr_hierarchy<xzarr_file_system_store> create_zarr_hierarchy(const std::string &
↪local_store_path, const std::string &zarr_version = "3")
```

Warning: doxygenfunction: Unable to resolve function "xt::get_zarr_hierarchy" with arguments None in doxygen xml output for project "xtensor-zarr" from directory: ../xml. Potential matches:

```
- template<class store_type> xzarr_hierarchy<store_type> get_zarr_hierarchy(store_
↪type &store, const std::string &zarr_version = "")
- xzarr_hierarchy<xzarr_file_system_store> get_zarr_hierarchy(const char *local_store_
↪path, const std::string &zarr_version = "")
- xzarr_hierarchy<xzarr_file_system_store> get_zarr_hierarchy(const std::string &
↪local_store_path, const std::string &zarr_version = "")
```

2.3.2 Store

Defined in `xtensor-zarr/xzarr_file_system_store.hpp`

class `xt::xzarr_file_system_store`

Zarr store handler for a local file system.

The *xzarr_file_system_store* class implements a handler to a Zarr store, and supports the read, write and list operations.

See *xzarr_hierarchy*

Public Functions

inline void **list_dir**(const std::string &prefix, std::vector<std::string> &keys, std::vector<std::string> &prefixes)

Retrieve all keys and prefixes with a given prefix and which do not contain the character "/" after the given prefix.

Parameters

- **prefix** – the prefix
- **keys** – set of keys to be returned by reference
- **prefixes** – set of prefixes to be returned by reference

inline std::vector<std::string> **list**()

Retrieve all keys from the store.

Returns returns a set of keys.

inline std::vector<std::string> **list_prefix**(const std::string &prefix)
Retrieve all keys with a given prefix from the store.

Parameters **prefix** – the prefix

Returns returns a set of keys with a given prefix.

inline void **erase**(const std::string &key)
Erase the given (key, value) pair from the store.

Parameters **key** – the key

inline void **erase_prefix**(const std::string &prefix)
Erase all the keys with the given prefix from the store.

Parameters **prefix** – the prefix

inline void **set**(const std::string &key, const std::string &value)
Store a (key, value) pair.

Parameters

- **key** – the key
- **value** – the value

inline std::string **get**(const std::string &key)
Retrieve the value associated with a given key.

Parameters **key** – the key to get the value from

Returns returns the value for the given key.

2.4 Build and configuration

2.4.1 Build the documentation

First install the tools required to build the documentation:

```
conda install breathe doxygen sphinx_rtd_theme -c conda-forge
```

You can then build the documentation:

```
cd docs
make html
```

Type `make help` to see the list of available documentation targets.

2.5 Releasing xtensor-zarr

2.5.1 Releasing a new version

From the master branch of xtensor-zarr

- Make sure that you are in sync with the master branch of the upstream remote.
- In file `xtensor_zarr_config.hpp`, set the macros for `XTENSOR_ZARR_VERSION_MAJOR`, `XTENSOR_ZARR_VERSION_MINOR` and `XTENSOR_ZARR_VERSION_PATCH` to the desired values.
- Update the readme file w.r.t. dependencies on xtensor.
- 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`)

2.5.2 Updating the conda-forge recipe

xtensor-zarr has been packaged for the conda package manager. Once the new tag has been pushed on GitHub, edit the conda-forge recipe for xtensor 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.

2.5.3 Updating the stable branch

Once the conda-forge package has been updated, update the `stable` branch to the newly added tag.

INDEX

X

xt::xzarr_file_system_store (*C++ class*), 10
xt::xzarr_file_system_store::erase (*C++ function*), 11
xt::xzarr_file_system_store::erase_prefix (*C++ function*), 11
xt::xzarr_file_system_store::get (*C++ function*), 11
xt::xzarr_file_system_store::list (*C++ function*), 10
xt::xzarr_file_system_store::list_dir (*C++ function*), 10
xt::xzarr_file_system_store::list_prefix (*C++ function*), 11
xt::xzarr_file_system_store::set (*C++ function*), 11
xt::xzarr_hierarchy (*C++ class*), 9