



Testing and testing infrastructure

Andrey Alekseenko

PDC, KTH Royal Institute of Technology, Stockholm

2023-09-08

Testing

- “In simple terms, it’s a process of checking something if it does what it intended to do”
<https://keencoder.dev/unit-testing-for-absolute-beginners>

Testing

- Unit tests: verify individual methods and functions
- Integration tests: verify that different modules work well together
- Functional tests: verify the final output of the application
- End-to-end tests: replicate a user behavior
- Acceptance testing: verify entire application for end goals
- Performance testing: verify reliability, speed, scalability
- Smoke testing: basic checks of major features

One of the possible nomenclatures; many others exists

Testing

- CI: practice of merging all developers' working copies to a shared mainline several times a day
- CD: software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, following an automated testing pipeline


Testing

- CI: practice of merging all developers' working copies to a shared mainline several times a day
- CD: software engineering approach in which teams ~~produce software in short cycles~~, ensuring that the software can be reliably released at any time and, following an automated testing pipeline
- Manual testing: not everything can be (easily) automated

GROMACS CI pipeline



Report NBNXM GPU supercluster dimensions in the log

 Open Szilárd Páll requested to merge `sz_allow_setting_and_repor...` into `main` 1 day ago

Overview 8 Commits 2 Pipelines 2 Changes 2

2 unreso

Also allow setting `DGMX_GPU_NB_NUM_CLUSTER_PER_CELL_X/Y/Z=1` with other backends too not just SYCL. The only case failing tests is `GMX_GPU_NB_CLUSTER_SIZE=4` on Intel (not on NVIDIA), but since we can't check at build-time for this, we forbid setting the cluster per cell in all cases when cluster size is 4.




 Merge request pipeline #994237396 passed

Merge request pipeline passed for `68e00fc2` 22 hours ago



8

Revoke approval

Requires 1 approval from GMX Developer Main. Approved by you 

 Test summary: **no** changed test results, **499** total tests

Full report

GROMACS CI pipeline



Pipeline Needs Jobs 55 Tests 499

Group jobs by Stage Job dependencies

pre-build

- clang-format
- copyright-check
- python-format
- simple-build

configure-build

- clang-tidy:configure-mr
- docs:configure
- gromacs:clang-9:configure
- gromacs:clang-13-mpi:configure
- gromacs:clang-ASAN:configure
- gromacs:clang-TSAN:configure
- gromacs:clang-UBSAN:configure
- gromacs:clang-static-analyzer:configure

build

- gromacs:clang-9:build
- gromacs:clang-13-mpi:build
- gromacs:clang-ASAN:build
- gromacs:clang-TSAN:build
- gromacs:clang-UBSAN:build
- gromacs:clang-static-analyzer:build
- gromacs:gcc-9-cuda-11.0.3:build
- gromacs:gcc-9-cuda-11.0.3:buildMPI

test

- gmxapi:clang-9:py-3.10
- gmxapi:clang-13-mpi:py-3.7
- gromacs:clang-9:regressiontest
- gromacs:clang-9:test
- gromacs:clang-13-mpi:test
- gromacs:clang-13:regressiontest
- gromacs:clang-ASAN:regressiontest
- gromacs:clang-ASAN:test

documentation

- docs:build

source-check

- check-source
- clang-tidy:test

post-test

- webpage:build

Parts of GROMACS automated testing

- Pre-build: code formatting, copyrights, simple build
- Configure-build: CMake in different configurations
- Build: Build the configurations above
- Test: Run the tests for the builds above
- Documentation: Check that documentation is building
- Source-check: Static analysis checks
- Post-test: Manual webpage can be built

Code style

- Clang-format: C++ code style
 - *Very* sensitive to clang-format version, needs to be 11.1.0
 - `sudo apt install clang-format-11`
- Python-format: Python code style (`black`)
- Copyright-check: Copyright headers

Code style



GROMACS > GROMACS > Jobs > #4985233672

```
at.sh --rev=$REV diff | tee clang-format.patch ; exit 1 ; +1
42 clang-format.sh found issues!
43 /// Patch below (can also be downloaded from artifacts) ///
44 diff --git org/api/nbllib/util/setup.cpp new/api/nbllib/util/setup.cpp
45 index b68d75a..03c5ea8 100644
46 --- org/api/nbllib/util/setup.cpp
47 +++ new/api/nbllib/util/setup.cpp
48 @@ -125,8 +125,7 @@ std::vector<Vec3> generateVelocity(real tempi, unsigned i
nt seed, std::vector<re
49  /*! (so the contrary of isfinite, see https://en.cppreference.com/w/cpp/numeric/math/isfinite)
50  bool isRealValued(gmx::ArrayRef<const Vec3> values)
51  {
52  -   return std::all_of(values.begin(), values.end(), [](const Vec3& val)
53  -   {
54  +   return std::all_of(values.begin(), values.end(), [](const Vec3& val) {
55         for (int m = 0; m < dimSize; ++m)
56         {
57             if (!std::isfinite(val[m]))
58         }
59  Uploading artifacts for failed job 00:02
60  Uploading artifacts...
61  clang-format.log: found 1 matching files and directories
    clang-format.patch: found 1 matching files and directories
```

only in main, not release-*

clang-format



Duration: 28 seconds

Finished: 1 week ago

Queued: 2 seconds

Timeout: 1h (from project) ?

Runner: #22213073 (Coj35txY9)

gitlab-runner-gitlab-runner-

c94b4dbff-cg7lj

Job artifacts ?

The artifacts will be removed in 3 weeks ?

Keep

Download

Browse

Commit 171f228c in !3782

Micro-optim isRealValued by using std::all_of + isfinite

✖ Pipeline #987039987 for !3782 with micro_optim

pre-build

Configure + Build + Test

`cmake && make && make check`

- Compiler: Clang/GCC
- MPI: on/off
- GPU: off/CUDA/OpenCL/SYCL
- Sanitizers: none/TSAN/ASAN/UBSAN

- “-test” and “-regressiontests”: two different test harnesses
- A few `gmxapi`-specific jobs

GROMACS CI pipeline



Pipeline Needs Jobs 55 Tests 499

Group jobs by

pre-build

- clang-format
- copyright-check
- python-format
- simple-build

configure-build

- clang-tidy:configure-mr
- docs:configure
- gromacs:clang-9:configure
- gromacs:clang-13-mpi:configure
- gromacs:clang-ASAN:configure
- gromacs:clang-TSAN:configure
- gromacs:clang-UBSAN:configure
- gromacs:clang-static-analyzer:configure

build

- gromacs:clang-9:build
- gromacs:clang-13-mpi:build
- gromacs:clang-ASAN:build
- gromacs:clang-TSAN:build
- gromacs:clang-UBSAN:build
- gromacs:clang-static-analyzer:build
- gromacs:gcc-9-cuda-11.0.3:build
- gromacs:gcc-9-cuda-11.0.3:buildMPI

test

- gmxapi:clang-9:py-3.10
- gmxapi:clang-13-mpi:py-3.7
- gromacs:clang-9:regressiontest
- gromacs:clang-9:test
- gromacs:clang-13-mpi:test
- gromacs:clang-13:regressiontest
- gromacs:clang-ASAN:regressiontest
- gromacs:clang-ASAN:test

documentation

- docs:build

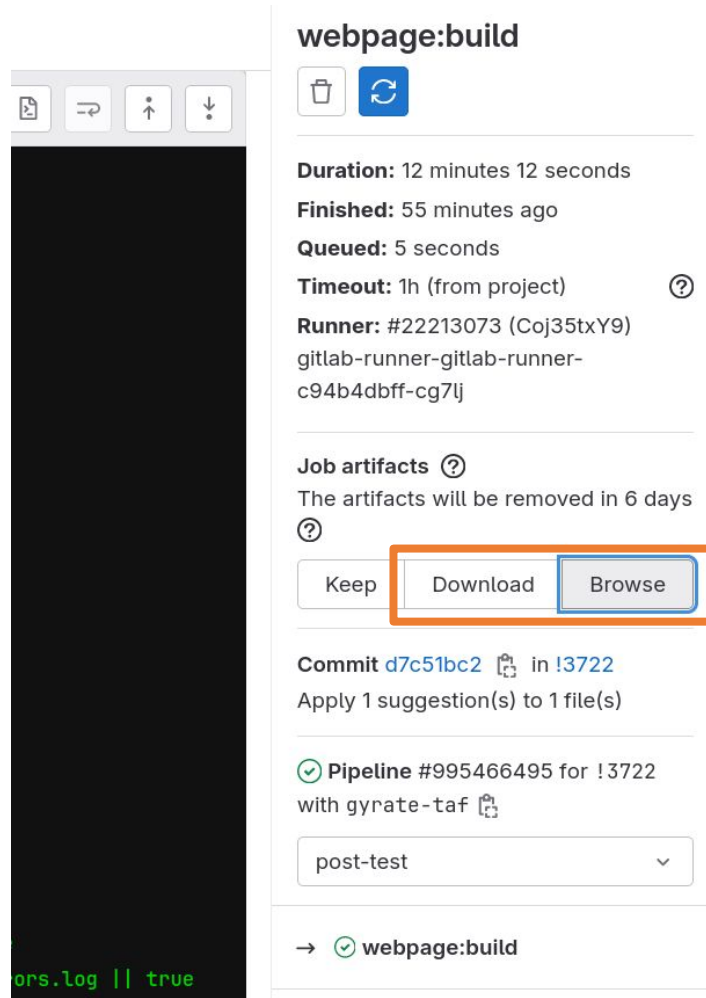
source-check

- check-source
- clang-tidy:test

post-test

- webpage:build

Documentation + webpage: preview



The screenshot shows a CI/CD interface for a job named 'webpage:build'. On the left, there is a dark terminal window with the text 'ors.log || true'. The main panel displays job metadata: 'Duration: 12 minutes 12 seconds', 'Finished: 55 minutes ago', 'Queued: 5 seconds', and 'Timeout: 1h (from project)'. The runner is identified as 'gittlab-runner-gittlab-runner-c94b4dbff-cg7lj'. Below this, the 'Job artifacts' section indicates they will be removed in 6 days and provides three buttons: 'Keep', 'Download', and 'Browse'. The 'Browse' button is highlighted with an orange border. Further down, the commit 'd7c51bc2' is shown, and the pipeline '#995466495' is listed. At the bottom, a dropdown menu shows 'post-test' and a navigation arrow points to the 'webpage:build' job.

Artifacts / build-docs / docs / html:

- index.html
- manual-2024-dev.pdf

Source-check

- check-source: basic documentation format checks
- clang-tidy: advanced code static analysis

Source-check

- check-source: basic documentation format checks
- clang-tidy: advanced code static analysis

```
if (i == 1 && i == 2) { // correct?  
}
```

Even more automated tests



- Post-merge (multi-GPU tests)
 - Heavy tests not to run on every minor change to the MR
- Nightly/weekly (exotic-ish hardware)
 - <https://gitlab.com/gromacs/gromacs/-/pipelines>
- GitHub actions (Windows, macOS)
 - <https://github.com/gromacs/gromacs/actions>

Not covered by automated testing

- Performance testing
- Many analysis tools
- Large-scale runs (> 2 GPUs, > 4 ranks)
- Rare devices (POWER9, ARM, high-end GPUs, etc)
- Long-running physical validation tests

Test frameworks

- GoogleTest for most tests
 - Modern and convenient
 - We call them “unit tests”, but that’s not always the case
- Old Perl scripts for regression tests
 - `cmake -DREGRESSIONTEST_DOWNLOAD=ON`
 - Don’t touch it unless you’re changing `mdrun` behavior :)
 - <https://gitlab.com/gromacs/gromacs-regressiontests/>

Running tests

- All tests:
 - `make tests && make check`
 - `ctest .`
- Specific test set:
 - `ctest -R MdrunIOTests`
 - `./bin/mdrun-io-test`
- Specific test case:
 - `./bin/mdrun-io-test --gtest_filter=GromppTest.*`
- Run multiple times (flaky test):
 - `ctest --repeat-until-fail 100 --output-on-failure -R MdrunIOTests`

Finding tests

- File: `src/gromacs/utility/logger.cpp`
- Tests: `src/gromacs/utility/tests/logger.cpp`
- Reference data: `src/gromacs/utility/tests/refdata/`
- Also in: `src/programs/*/tests/`

Understanding tests

```
TEST(EnergyTermTest, AddFrameWorks)
{
    EnergyTerm term(0, true, "test", "test");
    term.addFrame(2, 1000, 10, 50, 5, 255);
    term.addFrame(4, 2000, 10, 100, 10, 155);
    EXPECT_EQ(term.numFrames(), 2);
    auto errorEstimate = term.errorEstimate(1);
    ASSERT_TRUE(errorEstimate.has_value());
    EXPECT_REAL_EQ(errorEstimate.value(), 0);
}
```

Understanding tests

```
TEST_P(HbondModuleTest, Works)
{
    const auto params = GetParam();
    std::string name = std::get<0>(params);
    int startingValue = std::get<1>(params);
    // Do something with name and startingValue
}
```

```
INSTANTIATE_TEST_SUITE_P( //...
    ::testing::Combine(::testing::Values("name1", "name2"),
        ::testing::Range(4, 8, 1)), // ...
```

Reference data

```
TEST_F(WrapperTest, WrapsCorrectly)
{
    std::vector<std::string> wrapped = doThings();
    checker().checkSequence(wrapped.begin(), wrapped.end(), "Wrapped");
}
```

checker() will compare with data in
src/gromacs/*/tests/refdata/WrapperTest_WrapsCorrectly.xml

```
$ ./bin/utility-test -ref-data update-all
```

Writing tests

- When fixing a bug: write a test first
 - Helps replicating
 - Helps verifying that your fix works
 - Helps prevent it from happening again
- When adding a new feature: think first about how to test it
 - Test behavior, not implementation
- Some things are very hard to test. Such is life.

Further information

- Testing in GROMACS:
 - <https://manual.gromacs.org/current/dev-manual/testutils.html>
- More on reference data:
 - https://manual.gromacs.org/current/doxygen/html-lib/page_refdata.xhtml
- GoogleTest Primer:
 - <https://google.github.io/googletest/primer.html>