GROMACS - Task #2756

Feature # 2585 (Resolved): Infrastructure supporting external API

gmxapi integration testing

11/14/2018 09:11 AM - Eric Irrgang

Status:	In Progress	
Priority:	Normal	
Assignee:		
Category:	gmxapi	
Target version:	2021-refactoring	
Difficulty:	uncategorized	

Description

Supports gmxapi milestone 3 at parent issue.

Integration testing

GROMACS 2019 defines gmxapi 0.0.7. The kassonlab/gmxapi and kassonlab/sample_restraint repositories are the principal client software packages to test against a gmxapi compatible GROMACS installation. These packages are probably compatible with GROMACS 2020, but are not likely to be tested in the GROMACS project infrastructure.

GROMACS 2020 coincides with the definition of gmxapi 0.1, which is not fully compatible with GROMACS 2019, and probably will not be.

GROMACS infrastructure will include full gmxapi testing on GitLab Runner, but not on Jenkins.

Goals

- Gmxapi interfaces should continue functioning with unchanged semantics for other GROMACS changes, or API level needs to be incremented according to semantic versioning.
- External projects need to be tested outside of the gromacs build tree to validate external interfaces of installation. Suggested external projects: Python package, sample_restraint, validation test suite.
- Tests should be clear about the API version they are testing, and we should test all versions that aren't unsupported (though we need a policy in this regard) and we can note whether new API updates are backwards compatible.
- Forward-compatibility testing: we should at least know whether we are breaking old client code and include release notes, regardless of policy
- ABI compatibility testing TBD. (should we test mismatched compilers and such?)
- Example code in documentation should be tested, where possible.

Matrix

Matrix axes contain the following.

- Python versions 3.5, 3.6, 3.7, 3.8
- MPI implementations mpich, mvapich, openmpi
- C++ compilers (all supported clang, gcc, intel...)
- GROMACS with MPI, thread-MPI, and no MPI
- GPU domain decomposition (relevant mostly for MD plugin testing, but possibly more)
- GROMACS floating point precision
- GROMACS branches master, release-2020, and beyond
- gmxapi 0.1, 0.2, and beyond

Versions tested

Nightly

For version information not specified in the matrix description, I think it is appropriate to test the most recent minor version or dependency version.

pre-commit

- Python patch version: default or most recent at time of configuration.
- Python package dependencies: minimum version described in package requirements.
- C++ dependencies (pybind, googletest, Eigen?): minimum version described in package requirements.
- cross verification (from master) with a release branch: oldest known compatible tag.
- cross verification (from release branch patch) with master: tip.

Tasks

- [ ] Define testing matrices for the various schedules (nightly, pre-commit, others?). Choose the sparse elements to test.
- [ ] Add "nightly-test" stage?
- [ ] Set up nightly rebuild of Python "venvs"s. Tagged artifacts? Some other caching mechanism?
- [ ] Generate initial pre-commit venvs (Keys: Python version, MPI version, and C++ toolchain.)
- [ ] Document update procedure for pre-commit venvs.
- [ ] Is there a GROMACS installation artifact already generated?
- [ ] What stage should client software be built and tested in?
- [ ] Extend .test-script-template for pytest with JUnit XML output.

Job dependencies

Apologies for the poor ASCII art...

```
build GROMACS -> install GROMACS -> build sample_restraint -> C++ unit tests
                                                  >-> pytest tests (MPI does not need as full cove
rage; could depend on other variables)
                                \-> install gmxapi -> non-MPI pytest tests
                                                  \-> MPI pytest tests
                                                  \-> acceptance tests(?)
```

The most expensive part here is the GROMACS build and install. The client software builds take a few seconds. The pytest unit tests take well under a minute. These timings may be comparable to the container launch or artifact handling, so it is probably sufficient to have a single stage to

1. install gmxapi
2. build (and install) sample_restraint
3-7. run the various tests.

The choice also depends whether there is a reason to have more stages versus wider stages, because we should plan to cross-check versions of the gmxapi Python package and versions of the sample code, at least until we are confident that we have well-characterized and acceptable compatibility guarantee and versioning API.

Related issues:

Related to GROMACS - Task #2912: C++ extension module for Python bindings	**Closed**
Related to GROMACS - Task #3033: Clean up and modernize googletest bundling a...	**In Progress**
Related to GROMACS - Bug #3111: sample_restraint testing should not download ...	**Closed**
Related to GROMACS - Task #3133: Cookiecutter for sample_restraint	**In Progress**
Related to GROMACS - Task #3014: gmxapi example Python scripts	**In Progress**
Blocks GROMACS - Task #3271: Adopt PEP-518 for Python package build system co...	**Resolved**
Blocked by GROMACS - Task #3272: Port complete CI testing to Gitlab	**New**
Blocks GROMACS - Bug #3408: Gmxapi* tests segfault in rpmbuild	**Fix uploaded**

Associated revisions

Revision 3fd14b5f - 12/10/2018 05:07 PM - Paul Bauer

Disable gmxapi by default

Due to outstanding issues with the integration testing and tests failing
with large number of ranks, the gmxapi default has been changed to not
be build. In Jenkins, all supported builds still are still set to build
with GMXAPI enabled.

Refs #2765, #2722, #2756

Change-Id: I2cc42c461edc206aaa30be6cac3db0a52ccae991

Revision 4ed5d8ed - 04/24/2019 03:12 PM - Eric Irrgang

Read GMXRC in gmxapi docker test environment.

Allows expected behavior for gmxapi/ci-... docker containers and ensures that GROMACS components available during image build are available when running.

Allows `gmx` binary to be easily discovered by Python wrappers.

Ref: #2756

Gerrit patch set 9390/4

Revision 354da806 - 04/24/2019 03:31 PM - Eric Irrgang

More Python testing infrastructure for gmxapi.

- Add and improve test fixtures.
- Generate MD input files for tests.

Refs: #2756
Refs: #2896

Gerrit patch set 9392/2

Revision 929343d1 - 05/09/2019 01:12 PM - Eric Irrgang

More Python testing infrastructure for gmxapi.

- Add and improve test fixtures.
- Generate MD input files for tests.

Refs: #2756
Refs: #2896

Change-Id: I15ac8e44844cbf699cfe362e2fa443d07a355b3d

Revision b67801d7 - 06/18/2019 07:34 PM - Eric Irrgang

Updates to gmxapi docker entry points.

- Read GMXRC in gmxapi docker test environment. Allows `gmx` binary to be easily discovered by Python wrappers. Supports expected behavior for gmxapi/ci-... docker containers and ensures that GROMACS components available during image build are available when running.
- Allow more granular testing. Rename and reorganize entry points for the docker test containers.

Ref: #2756

updates from new patch set to change 9390

Revision e5a8e153 - 06/19/2019 11:38 AM - Eric Irrgang

Updates to gmxapi docker entry points.

- Read GMXRC in gmxapi docker test environment. Allows `gmx` binary to be easily discovered by Python wrappers. Supports expected behavior for gmxapi/ci-... docker containers and ensures that GROMACS components available during image build are available when running.
- Allow more granular testing. Rename and reorganize entry points for the docker test containers.

Ref: #2756

Change-Id: I5e781fae4709e4ccb718a8295c3fbe68dd59de44

Revision 42adebaf - 07/24/2019 06:19 AM - Eric Irrgang

Improve gmxapi docker testing environment.

- Generalize venv directory.
- Add some usage notes.
- Better support alternative users in executing containers.
- Remove stale TODO.
- Allow for passing additional arguments to pytest.

Refs #2756

Change-Id: I9f67d46aa93fb789825cb78cc2a95b24f7dfccaa

Revision 7cf28864 - 07/24/2019 11:58 AM - Eric Irrgang

Clarify use of Python venv in gmxapi docker images.

Refs #2756

Change-Id: I182c83be424fcf134bc0f3ce7c41a58c6fb1f07e

Revision 6ec0f1f1 - 08/08/2019 08:19 PM - Eric Irrgang

Allow gmxapi Python package tests to be run from GROMACS build tree.

- Configure gmxapi._gmxapi to build with the ability to find libgmxapi.
- Add a gmxapi_pytest custom CMake target to invoke pytest.

Refs #2961
Refs #2756

Change-Id: I64ca82afbf37da3c37759ec302c2ac9cc1666de2

Revision 4217d8fc - 08/21/2019 03:37 PM - Eric Irrgang

Enable GMXAPI by default.

Change default CMake configuration to enable the build and install of
libgmxapi and associated headers.

Refs #2756
Refs #3059

Change-Id: I5688a1ad6b524882090201014fc34cb34c1e3748

Revision de509583 - 03/31/2020 07:50 PM - M. Eric Irrgang

Prepare Python environments for GitLab CI pipeline.

- Install Python interpreters.
- Prepare and stash Python virtual environments.

Refs #2756

Change-Id: I1b68e039018a72865d1cc6f60557104026328285

Revision 5d3cbc77 - 04/07/2020 01:17 PM - M. Eric Irrgang

Prepare Python environments for GitLab CI pipeline.

- Install Python interpreters.
- Prepare and stash Python virtual environments.

Refs #2756

Change-Id: I1b68e039018a72865d1cc6f60557104026328285

History

#1 - 11/14/2018 09:12 AM - Eric Irrgang

- Tracker changed from Feature to Task

#2 - 11/16/2018 01:50 PM - Eric Irrgang

- Blocks Bug #2722: gmxapi may over-manage RPATH added

#3 - 12/07/2018 12:05 PM - Gerrit Code Review Bot

Gerrit received a related patchset '7' for Issue #2756.
Uploader: Paul Bauer (paul.bauer.q@gmail.com)
Change-Id: gromacs~release-2019~I9ad2995263c7798c2c4569377be701d4a8ec7291
Gerrit URL: https://gerrit.gromacs.org/8667

#4 - 12/10/2018 02:03 PM - Paul Bauer

I spend some time on this now, and while I have a setup that works on my machine it seemed to time out on the Jenkins agent I was running it on.
I also have some reservations to the CMake setup, as we don't need to test e.g. the documentation generation during our Jenkins builds.
Additionally, I see an issue with race conditions when two different builds try to build the tests at the same time and one of them spontaneously decides to uninstall the installed version why another one is running tests at the moment.

#5 - 12/10/2018 02:04 PM - Paul Bauer

Also, the CI scripts don't seem to support versioned builds, making it difficult to have proper version testing in Jenkins

#6 - 12/10/2018 04:03 PM - Gerrit Code Review Bot

Gerrit received a related DRAFT patchset '1' for Issue #2756.
Uploader: Paul Bauer (paul.bauer.q@gmail.com)
Change-Id: gromacs~release-2019~I2cc42c461edc206aaa30be6cac3db0a52ccae991
Gerrit URL: https://gerrit.gromacs.org/8801

#7 - 12/17/2018 01:32 PM - Paul Bauer

- Target version changed from 2019 to 2020

More likely to be done in 2020

#8 - 12/19/2018 12:58 AM - Szilárd Páll

- Category set to analysis tools

As noted on gerrit, I think such proposals are better suited for an earlier release stage rather than in the middle of the betas.

Secondly, I'm not certain that testing GROMACS' own API with/against external software is the best choice. Why not have API integration tests implemented in the source repo?

#9 - 12/19/2018 10:14 AM - Erik Lindahl

Agree about testing location. If the API is part of the source tree, the full integration tests should also live there so anybody can test it while developing, not in a separate external repo.

#10 - 12/19/2018 11:31 AM - Eric Irrgang

> better suited for an earlier release stage

To be clear, this has been part of ongoing discussion since before #2585. I created this task issue to collect notes from private conversations and to have something to link in the Gerrit changes that started appearing.

> If the API is part of the source tree, the full integration tests should also live there so anybody can test it while developing

The API is tested in the source repo as much as possible, but the C++ API is limited enough right now that it is hard to write thorough tests without developing an entire application. A big part of the test space targeted here is the extensibility.

The current GROMACS testing infrastructure gives access to the GROMACS build tree and libgromacs, so it is not suited to testing client software. I think we should work on infrastructure in the GROMACS repo to allow more isolated testing, but I think that is somewhat of a separate issue.

For historic reasons, it is hard to tell whether there are behaviors in GROMACS that might change unexpectedly and violate an assumption that was based on observation rather than specification.

This issue (#2756) was intended to be about using a client software package whose behavior we control and understand to check for surprises when built against a GROMACS installation. Currently, the only (known) client software are the two packages I've proposed using.

I'm not sure it is well documented yet, but recent discussion indicates that we should plan to test gmxapi 0.0.7 code against GROMACS 2019, though gmxapi 0.0.8 client code will likely build against both GROMACS 2019 and later. We may want to fork the discussion to address (a) tests on the 2019 release branch, and (b) tests on master and 2020.

#11 - 12/20/2018 10:24 AM - Paul Bauer

Here is the follow-up from the dev teleconference

Things to consider for getting integration tests running during Jenkins verification

- Rework build script to have gmx-integration build target that only builds what is necessary for tests to run and complete

- Consider requirement for C++14 in gmxapi build (just remembered this as well)
- Find way to reuse virtual envrionment -> Is it possible to archive a previously constructed venv and unpack it?
- Work around requirement to know C compiler version used for mpi4py (as instructed in build script: CC holds the compiler that is also associated with the `mpicc` used to build the mpi4py Python package.)
- Lay out plan to bind current gmxapi closer to general GROMACS API planning and integration to remove need for separate repository (can still be upstream, but should be included in main tree)

#12 - 03/02/2019 01:10 AM - Eric Irrgang

This testing is slated to migrate to Docker-based CI infrastructure. Until that is set up on Jenkins, external relevant CI testing is visible at https://travis-ci.org/kassonlab/gromacs-gmxapi/branches

#13 - 03/31/2019 04:47 PM - Gerrit Code Review Bot

Gerrit received a related DRAFT patchset '4' for Issue #2756.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~Ic74ee9abc70f9d1be3e1d94619477c65ee093cb0
Gerrit URL: https://gerrit.gromacs.org/9336

#14 - 03/31/2019 05:47 PM - Eric Irrgang

- Related to Task #2912: C++ extension module for Python bindings added

#15 - 04/02/2019 10:35 AM - Eric Irrgang

A note to consider test data availability: src/testutils/simulationdatabase will ought to be sufficient for C++ API tests. Presumably client code of the C++ API (such as the Python package) should also use the standardized test data from this source. Should the data be installable with GROMACS? Available through the C++ API? Copied into the Python packaging area by the top-level CMake project? I could imagine CMake flags like "INSTALL_TEST_DATA" or pytest arguments like "--test-data-directory".

#16 - 04/02/2019 10:38 AM - Eric Irrgang

Possibly related to https://redmine.gromacs.org/issues/2795

#17 - 04/03/2019 11:16 AM - Gerrit Code Review Bot

Gerrit received a related DRAFT patchset '1' for Issue #2756.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I5e781fae4709e4ccb718a8295c3fbe68dd59de44
Gerrit URL: https://gerrit.gromacs.org/9390

#18 - 04/03/2019 12:57 PM - Gerrit Code Review Bot

Gerrit received a related DRAFT patchset '1' for Issue #2756.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I15ac8e44844cbf699cfe362e2fa443d07a355b3d
Gerrit URL: https://gerrit.gromacs.org/9392

#19 - 05/14/2019 04:23 PM - Eric Irrgang

Updates from discussion with Mark this morning regarding new Jenkins pipelines implemented as Kubernetes pods.

Tool chains

New testing infrastructure will provision build hosts with single toolchains so that compiler hinting is not necessary. This means that we won't have to do any extra checking to make sure that mpicc discovered by CMake or pip are compatible with the default compilers or with those used by different components (libgromacs vs. mpi4py, etc.).

Driving the builds

For convenience, all building and testing will be driven by CMake and CTest. We can build the package in the build tree with CMake and then invoke pytest from a CTest target with an appropriate PYTHONPATH.

Processing test results consists of evaluating the return value of the CTest and, optionally, scripted processing of console output or JUnit XML output as a filesystem artifact tracked by Jenkins.

Coverage and optimization

To better simulate real-world use cases, we should have a CMake build target that either builds a virtualenv or receives one as an argument, then runs pip install for the package.
We can configure the Docker images for the build hosts to have a Python virtualenv pre-configured for the jenkins user with the minimum versions of the Python package dependencies. This virtualenv could be activated as part of the environment set up for the Jenkins agent or passed as a special CMake flag. We can also/instead test --user installs for system Python installations or non-virtualenv installs for user space Python installations.

In any case, with the environment pre-configured, we can run pip with --no-index to build and install the package with baseline dependencies.

We should also test up-to-date Python packages (at least occasionally) either by periodically preparing an environment with then-current packages or by having a Jenkins test that allows PyPI to resolve the package dependencies. Note that we can also stash a bunch of downloaded packages in a directory to reference with the pip install --find-links option.

#20 - 07/04/2019 10:54 AM - Eric Irrgang

- Status changed from New to In Progress

- Target version changed from 2020 to 2020-infrastructure-stable

Note that, in addition to figuring out how to give the sample_restraint tree access to pybind11 and googletest sources, we need to tell it how to find the gmxapi package in the build tree (${CMAKE_BINARY_DIR}/python_packaging/src/gmxapi_staging) when running the Python-based tests. We also need CTest wrappers for both packages.

#21 - 07/17/2019 05:10 PM - Eric Irrgang

- Related to Task #3033: Clean up and modernize googletest bundling and usage added

#22 - 08/16/2019 09:56 AM - Mark Abraham

- Target version changed from 2020-infrastructure-stable to 2020-infrastructure-update-post-beta1

#23 - 10/13/2019 12:30 PM - Eric Irrgang

- Related to Bug #3111: sample_restraint testing should not download files added

#24 - 10/13/2019 01:07 PM - Eric Irrgang

- Related to Task #3133: Cookiecutter for sample_restraint added

#25 - 10/29/2019 05:39 PM - Paul Bauer

- Target version changed from 2020-infrastructure-update-post-beta1 to 2020-beta3

bumped to beta3

#26 - 12/02/2019 09:46 AM - Paul Bauer

- Target version changed from 2020-beta3 to 2020-rc1

I'll bump this for now to rc1
@Eric, what is still missing here?

#27 - 12/03/2019 06:20 PM - Eric Irrgang

Paul Bauer wrote:

> I'll bump this for now to rc1
> @Eric, what is still missing here?

As far as I know, the main thing missing is a CI environment.

A minimal resolution could consist of

- running the gmxapi_pytest and gmxapi_extension_pytest targets for commits affecting the installed headers, and
- building, installing, and testing the client Python packages nightly with a few different environments.

#28 - 12/11/2019 11:30 AM - Eric Irrgang

- Related to Task #3014: gmxapi example Python scripts added

#29 - 12/11/2019 12:44 PM - Eric Irrgang

If there is a tracked issue for the GitLab Runner infrastructure, we could make this issue dependent on it and update the target version to match. Otherwise, I need to defer to Paul on reasonable target version.

#30 - 12/11/2019 01:49 PM - Paul Bauer

Hello Eric, please check out https://gitlab.com/gromacs/gromacs-testing/-/jobs/376045509 and the corresponding pipeline for a build and test with PYTHON_PACKAGING=ON :)

#31 - 12/18/2019 06:02 PM - Eric Irrgang

Paul Bauer wrote:

> Hello Eric, please check out https://gitlab.com/gromacs/gromacs-testing/-/jobs/376045509 and the corresponding pipeline for a build and test with PYTHON_PACKAGING=ON :)

Great!

I see that the pipeline invokes ctest directly rather than calling make check. It also looks like there is JUnit XML processing going on. As such, the gmxapi_pytest and gmxapi_extension_pytest cmake targets probably aren't as useful as invoking pytest directly. I can help set that up, if that's the way you want to go. I'll try to prepare a change on top of yours.

#32 - 12/20/2019 08:21 AM - Paul Bauer

- Target version changed from 2020-rc1 to 2021-infrastructure-stable

I changed this so it no longer blocks the RPATH bug and has a target for next year

#33 - 12/20/2019 08:22 AM - Paul Bauer

- Blocks deleted (Bug #2722: gmxapi may over-manage RPATH)

#34 - 12/25/2019 01:37 PM - Eric Irrgang

- Description updated

#35 - 12/25/2019 01:37 PM - Eric Irrgang

- Category changed from analysis tools to gmxapi

#36 - 01/16/2020 04:51 PM - Eric Irrgang

- Blocks Task #3271: Adopt PEP-518 for Python package build system configuration. added

#37 - 01/16/2020 05:12 PM - Eric Irrgang

- Blocked by Task #3272: Port complete CI testing to Gitlab added

#38 - 01/16/2020 05:12 PM - Eric Irrgang

- Target version changed from 2021-infrastructure-stable to 2021-refactoring

#39 - 03/06/2020 04:01 PM - Eric Irrgang

- Blocks Bug #3408: Gmxapi tests segfault in rpmbuild added*