GROMACS - Feature #2585

Infrastructure supporting external API

07/24/2018 03:52 PM - Eric Irrgang

Status:	Resolved
Priority:	Normal
Assignee:	
Category:	
Target version:	
Difficulty:	uncategorized

Description

This issue describes the major feature developments necessary to support development of external API facilities with appropriate abstraction and integration testing. With issues like #988 still open, it is still possible to describe necessary GROMACS infrastructure changes to support API development and the best known API functionality goals.

Goals

The following list summarizes feature requirements established during development of https://github.com/kassonlab/gmxapi

- MPI environment sharing
- Binding external objects with interfaces for external code to retain access to launched tasks (such as objects instantiated during thread-MPI launch)
- Accessible simulation stop signal
- Launch MD runner through library calls
- Extensible ForceProviders
- Restraint framework / extensible “pull” code
- Libgmxapi build target and public headers
- Libgmxapi doxygen documentation
- Manage filesystem input and output locations
- Interaction / synchronization with predetermined integer time steps rather than time value.

Packaging and project structure

The following notes may belong in a different issue, such as #2045, but are provided here for context.

Libgromacs

Long term expectations consistent with this proposal:

- Its modules may have headers internal to them,
- Exposes a library API of deliberately selected components
- Its modules may expose components to that library API e.g. IForceProvider,
- That library API must be unit tested so it is known to work
- does not expose a user-facing API (ie no installed headers)
- Library API will be stable over the lifetime of a release branch (ie from about the time of the official release),
- Soversion bumps with each major release
- Library API will not be stable in master branch (but must pass its own tests in CI)
- Expects to be able to link itself dynamically
- Expects to be able to link itself statically (and its dependencies, if available)
- Linked as position-independent code
- Requires python v3 (patch from Pascal in gerrit, also releng needs an update too, probably)

Libgmxapi in GROMACS repo

While the exposed symbols and public headers are reconsidered and as code in libgromacs continues to undergo more compartmentalization and modernization, it is convenient to define a second library and set of public headers. For now, we defer the question of whether libgmxapi would ultimately be subsumed into libgromacs, but we propose that the gmxapi headers ultimately replace the current set of public headers.

- Builds with access to libgromacs library API headers
- installs the user-facing C++ headers for client code (such as what Python module builds against)
- “plugin” API. E.g. a stable framework(s) for implementing various sorts of potentials while hiding IForceProvider e.g. by exposing adapters to things like domain-distributed positions. May warrant different versioning semantics than API/ABI supporting user interfaces: probably more stable API; possibly less stable ABI.
- (Proposing) semantic versioning (standard guarantees starting somewhere between 0.1 and 1.0) https://semver.org/
- One name or separate name for tMPI / MPI? (suggest “no”)
- Plugin API only in C++, but has supported method to expose bindings to the inserted code that can be used in Python scripts, i.e plugins are not written in pure Python
- Main (high-level workflow) API complete only in python, but will migrate to C++. (We can accelerate the C++ version if there is a need)
- No plans to load C++ plugin modules natively (ie no dlopen) because this is able to be handled more portably at Python level (or some other user interface code), but we need to be able to receive and register factory functions to get objects from external code

Parts of this discussion may belong under #988

Python package

A GROMACS Python package like https://github.com/kassonlab/gmxapi is nominally beyond the scope of this issue. A few points from #2045 worth mentioning, though:

- serves to prove libgmxapi functionality; needs integration testing with libgromacs and libgmxapi
- Ultimately, it should be available as part of a GROMACS installation, but should it be in the same repository and/or CMake build environment?
- pybind11 source bundled with repo
- Provides both an implementation and an API specification.
  - Researchers can be compatible with it without depending on it.
  - Writers of GROMACS extension code are free to use other Python bindings frameworks.
  - We provide tools and helpers, but C++ helpers use pybind11

Details

A big picture of planned development is necessary even before Redmine Issues exist, so milestones are enumerated with feature ID tags. Dependencies are better illustrated in the accompanying chart. Each numbered feature (chart node) is expected to be from 1 to 10 Gerrit changes, generally 3 to 5.

Proposed development targets

Gmx1 (this issue) Design documentation strategy / project management plan

- this issue and a cluster of Redmine issues with subtasks should be good
- documentation and visual aids like the attached progress chart should probably be in the repository somewhere

Gmx2 (Issue #2586) Versioned libgmxapi target for build, install, headers, docs

Gmx3 Integration testing (Issue #2756)

- 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, yet-to-be-written integration 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? (should we test mismatched compilers and such?)
- Example code in documentation should be tested, if possible.

Gmx4 Library access to MD runner

- mdrun CLI program is an API client

Relates to #2229

Gmx5 (Issue #2587)Provide runner with context manager

Gmx6 Extensible MDModules and ForceProviders

- ForceProviders obtained after tMPI threads have spawned.
- MDModules list extended at runtime during simulation launch.
- External code may be provided to the runner to instantiate or get a handle to a module.
- Expanded Context class can broker object binding by registering and holding factory functions for modules, as well as other resources that may be implemented differently in different environments.
- Somewhere in here, MDModules either need access to the integral timestep number or the ability to register call-backs or signals on a schedule.

Relates to #2590, #2574, #1972

Do MDModules live in a scope of tight association with an integrator? Do we need other concepts, like RunnerModules? Or subdivisions like MDForceModule, MDObserverModule, MDControlModule?

Gmx7 Binding API for higher-level client code

Gmx8 Binding API for plug-in ForceProviders

Ultimately tied to gmx5 and gmx24, but we can start stabilizing the external interfaces now. The external interfaces are for (a) user interface / workflow management code, and (b) MD extension code. We define a simple message-passing C structure along with PyCapsule name and semantics. An MD extension object can provide a factory method with which the MD Runner can get an IMDModules interface at simulation launch. The object pointed to may exist before and/or after the lifetime of the simulation. It must be understood that the IMDModule handle will be obtained on every rank. Design should consider future infrastructure and needs, but does not need to implement now. (expressing data dependencies and locality, negotiating parallelism, expressing periodicity) Short-term implementation may require workarounds for some of these, but the workaround can mostly be segregated from this issue's resolution.

Relates to #2590

Gmx9 Headers and adapter classes for Restraint framework

Relates to #1972, #2590, https://github.com/kassonlab/sample_restraint

Gmx10 MD signalling API

Relates to #2224

Gmx11 Replace MPI_COMM_WORLD with a Context resource

Gmx12 Runtime API for sharing / discovering hardware / parallelism resources

- Libgmxapi requests resources from libgromacs from the current node
- CUDA environment can be manipulated but we shouldn't have to deal with that for a while
- Evolving task scheduling interfaces, expressing data locality
- Concepts of time and timestep

Gmx13 API for working directory, input/output targets?

Gmx14 Generalized pull groups / "generalized sites"

Christian Blau actively working on this from mid-July

Gmx15 API logging resource

Log "file" artifacts are produced through API, allowing extensibility and abstraction from filesystem dependence. Progress has already been made in this direction, but the logging resource could be more clearly owned by the client code (or a Context object owned or managed on behalf of the client code) rather than created and destroyed in, say, the Mdrunner.

Also relates to #2570

Gmx16 Exception handler firewall

currently the gmx binary has a commandline runner thing that catches the exceptions, reports an error and exits, but the API can and should do something else, because it plays the same role as the commandline runner

Gmx17 API status object

- Status type defines the interface for discovering operation success or failure, plus details.
- Consistent status object interface is portable across Python, C++, and C
- Status object can be used to ferry information across API boundaries from exceptions thrown. Exceptions could be chained / status nested.

Questions:

- What are concerns and solutions for memory allocation for status objects? Should objects own one or generate one on function return?
- Should the API (or Context) keep a Status singleton? A Status stack? Or should operations create ephemeral Status objects, or objects implementing a Status interface?
- Should the status object contain strings, reference strings mapped by enum, or defer textual messages to messaging and logging facilities?

Gmx18 Thinner test harness (for API client tests)

Gmx19 API manipulation of simulation input / output

(for better testing) - GlobalTopology class and IGlobalTopologyUser interface underway will help here, so that client changes to the global topology can ripple through to the modules because the ones that care have registered themselves at setup time

Gmx20 Accessible test data resources

Gmx21 Break up runner state into a hierarchy of RAII classes with API hooks

- break up mdrun program into clearly defined layers and phases
- CLI program parses various inputs in order to launch an Mdrunner object that is CLI-agnostic
- launching tMPI threads and other significant changes of state establish a sequence or hierarchy of invariants through RAII and/or State pattern.
- Sebastian Wingbermuhle working now on aspects of this for hybrid MC/MD (ref #2375, …)

Gmx22 API management of input objects

- Structure, topology
- Microstate
- Simulation state
- Simulation parameters
- Runtime parameters / execution environment
- Anything else?

Gmx23 Event hooks or signals

Event hooks or signals for

- checkpoint
- time step number or delta / trajectory advancement
- input configuration
- input topology
- input state
- simulation parameters
- output data streams

Gmx24 API expression of MDOptions interfaces and embedded user documentation

Gmx25 Avoid sys::exit

Generally, replace std::exit (gmx_fatal)with exceptions

- Root out gmx_fatal, clearly define regular exit points and exception throwers
- API firewall should catch exceptions from gmx and convert to status objects for ABI compatibility. (gmx17)

- Clearly document regular and irregular shutdown behavior under MPI, tMPI, and generally, specifying responsibilities
- Create issue tickets for discovered missing exception safety, memory leaks, opportunities for RAII refactoring, and complicated protocols that should either be better documented or replaced with a clearer hierarchy (or sequence) of invariants

gmx26 API messaging resources

Abstraction for status messages, such as are currently printed to stdout or stderr

gmx27 (retracted)

gmx28 set simulation parameters from API

Short term: mdrun CLI-like functionality to override other input is sufficient

Long term: sufficient API to update parameters between phases of simulation work

Implementation roadmap is probably

1. Inject argv fields
2. Write to input_rec or other structures
3. Interact with MDOptions framework

gmx29 API access to grompp functionality

- Generate runnable input from user input
- United implementation for workflow API and utility functions (e.g. possibility of deferred execution / data transfer)
- Ultimately should not require writing output to (tpr) file
- File inputs ultimately should be generalized to API objects

gmx30 API access to GROMACS file manipulation and topology manipulation tools.

- United implementation for workflow API and utility functions (e.g. possibility of deferred execution / data transfer)
- Utility API should be sufficient to reimplement CLI tools
- I/O should ultimately be separate from algorithm; filesystem interaction optional
- Consider feature requirements of other projects such as MDAnalysis.

(Issue #2698) gmx31 Documentation integration.

Establish policies and layout for external (installed) API documentation, extension API for plug-in developers, and developer documentation for API and library implementation levels. Integrate with previous `webpage-sphinx` and `doxygen` targets and output.

Scope

There are definitely design points for consideration that are left out of this list merely because they are not essential to gmxapi functionality or because gmxapi doesn't have strong dependence on the ultimate design choice. These topics include:

- Task scheduling framework
- Insertion points in the MD loop
- Encapsulation of integrator

Further downstream, this infrastructure is necessary to support new high level interfaces to GROMACS, but the discussion of such interfaces is deferred as much as possible to separate issues to streamline incorporation of the changes proposed here in less public / stable code.

Criteria for Completion

This issue is resolved when sufficient infrastructure is in place to support ongoing development of the other subprojects in issue #2045 against the GROMACS master branch in Gerrit. In particular, the proof-of-concept client code at https://github.com/kassonlab/gmxapi and https://github.com/kassonlab/sample_restraint would not require a forked specialized copy of GROMACS.

Subtasks:

Feature # 2586: Versioned libgmxapi target for build, install, headers, docs	**Closed**
Feature # 2587: Provide Context (e.g. to runner code) to manage client and runtime envi...	**In Progress**

Feature # 2605: Library access to MD runner	**Closed**
Feature # 2610: API status object	**Closed**
Feature # 2620: MD signaling API	**Closed**
Task # 2623: Allow extensible MDModules and forceProviders.	**Closed**
Task # 2630: gmxapi integration testing	**Closed**
Feature # 2651: clean up mdrun log file handling	**Closed**
Task # 2756: gmxapi integration testing	**In Progress**

Related issues:

Related to GROMACS - Task #2045: API design and language bindings	**New**
Related to GROMACS - Feature #2229: Full Object Oriented Modularization of GR...	**New**
Related to GROMACS - Task #2590: Essential Dynamics as module providing forces	**New**
Related to GROMACS - Feature #2574: iForceSchedule Abstraction	**New**
Related to GROMACS - Feature #1972: external potential modules for refinement...	**New**
Related to GROMACS - Feature #2224: Proposed feature: conditional stop	**New**

Associated revisions

Revision 9864b201 - 08/29/2018 02:32 PM - Eric Irrgang

Allow extensible MDModules and forceProviders.

supports gmxapi milestone 6, described at #2585.

MDModules::Impl gets a std::vector for (shared) ownership of objects
providing the IMDModule interface. An add() method is added to the
MDModules public member functions, but the binding protocols are
separated into separate issues to allow minimal changes and varying
dependencies on other pending changes.

Relates to #1972, #2229, #2492, #2574, #2590.

Refs #2623

Change-Id: Ibb16d1453003213a49622810ed8bad4ed4b06e2d

Revision 6c9906e6 - 09/23/2018 01:59 PM - Eric Irrgang

Add gmxapi::MDModule.

Provide basic framework for passing objects between object libraries.
Describe protocol for managing object lifetimes and sharing ownership
across API environments (such as Python, C, and C++) with minimal
dependencies.

Supports gmxapi milestone 7:

Refs #2585

Change-Id: I206adb7341b3005dd56137e7c7df207a08dd41c2

Revision 24e70bfa - 10/03/2018 12:07 PM - Eric Irrgang

Add test data for gmxapi.

Build input file for a small system for basic gmxapi testing.

These test files either come from the mdrun tests or from manipulation
of that data. With access to test data and API tools to manipulate
input configurations, these files can be moved or removed.

Relates to gmxapi milestone 20 as described in issue #2585

Change-Id: I0eeb1d15120d3eb5309ec38c57e4e6f60097885a

Revision 1b34724f - 10/11/2018 11:57 AM - Eric Irrgang

Add gmxapi::Session.

Allow API client to launch simulation work in a provided Context.
Supports running a simulation from a TPR file.

Support gmxapi milestone 7

Refs #2585

Change-Id: Ie57962a739ae722b1ced8dd75f0037f07be3e1fa

Revision 5327f266 - 10/11/2018 12:23 PM - Eric Irrgang

Add gmxapi::Workflow.

Hold a description of simulation work to be performed. Work description
is a dependency graph that allows a Context implementation to configure
and launch an API Session to satisfy data flow constraints. Initial
implementation just provides command-line type input for mdrun in the
form of a TPR filename. A more complete implementation would look more
like the "work specification" described in
https://doi.org/10.1093/bioinformatics/bty484

For more discussion on a full implementation, refer to
https://github.com/kassonlab/gmxapi/issues?q=milestone%3Agmxapi_workspec_0_2+

Refs #2585

Change-Id: I2c9bb99d88a11893201489757453ef25bc35a52

Revision 6f025f6e - 10/12/2018 10:08 AM - Mark Abraham

Exclude gmxapi source files from include sorting

For now, we are taking no care of such aspects, so the attributes
instruct the include sorter to ignore matching files. The check-source
target suppresses such files in other ways.

Refs #2585

Change-Id: Ib84a89bfa7d5eb54a5a3585d65725fedf98fbab4

Revision 0c6b7e5a - 10/12/2018 01:58 PM - Eric Irrgang

Add restraints framework.

Extension code can implement IRestraintPotential and register to
provide MD forces and energy. Client code is not necessarily expected
to implement the interface directly, but likely with the help of
template headers.

Restraint functors should ultimately be retrieved or instantiated from
factory functions in the API Context manager provided to the simulation,
but in the current implementation, registered restraints are made
available during launch through the MdrunnerBuilder.

For sample extension code, see
https://github.com/kassonlab/sample_restraint

Supports gmxapi milestone 9,

Refs #2585

Change-Id: I48d57570721223eacca957c9a0f7ae98163ab6ff

Revision a888e947 - 10/12/2018 06:50 PM - Eric Irrgang

Add API for adding MDModules / Restraints.

Add gmxapi::MDWorkSpec, which can be passed through MDHolder.
It is a near-term solution until an API Context manager can be
more fully realized. A future implementation should instead
register factories for simulation support code in a Context
that is passed from the client. This would resolve ambiguity
about the purpose of gmxapi::MDWorkSpec versus the more
general gmxapi::Workflow, but in the present form, MDWorkSpec
is a public API class, while Workflow is still an implementation
detail.

Supports gmxapi milestone 7.

Refs #2585

Change-Id: I41fb13646c41b0f6e7690a0cedabd17eb3c47433

Revision 8bc1688d - 10/12/2018 10:22 PM - Eric Irrgang

Enable restraints facilities.

Extend gmxapi to allow a client to attach code to an MD simulation
through the restraints framework. Attach any registered restraints
during session launch.

Support gmxapi milestones 7 and 9, described in

Refs #2585

Change-Id: I61f7b52f1490e7692cdf7f14c7f46f8970b1c30a

Revision 7814d727 - 10/15/2018 10:41 AM - Mark Abraham

De-duplicating mdrun setup

Made the API context setup use LegacyMdrunOptions to keep the two
client setup procedures in step and avoid duplicating code.

Refs #2585

Change-Id: I9909e26a41fd2015667c2f86335c09b6f9708e99

Revision 7347b922 - 10/15/2018 02:22 PM - Eric Irrgang

Release notes for restraint module and gmxapi.

Describe new features supporting gmxapi client code and runtime binding
of restraint code. Add "gmxapi external API" section to install guide.

Refs #2585

Change-Id: I3da0d130b07862abc90387ade781d111ccbcd740

History

#1 - 07/24/2018 03:57 PM - Eric Irrgang

- Related to Task #2045: API design and language bindings added

#2 - 07/30/2018 05:43 PM - Eric Irrgang

- Description updated

#3 - 07/30/2018 05:47 PM - Eric Irrgang

- Related to Feature #2229: Full Object Oriented Modularization of GROMACS MDRUN Codebase added

#4 - 07/30/2018 05:47 PM - Eric Irrgang

- Related to Task #2590: Essential Dynamics as module providing forces added

#5 - 07/30/2018 05:47 PM - Eric Irrgang

- Related to Feature #2574: iForceSchedule Abstraction added

#6 - 07/30/2018 05:47 PM - Eric Irrgang

- Related to Feature #1972: external potential modules for refinement against experimental data added

#7 - 07/30/2018 05:49 PM - Eric Irrgang

- Related to Feature #2224: Proposed feature: conditional stop added

#8 - 07/30/2018 05:50 PM - Eric Irrgang

- Subject changed from Infrastructure to support external API to Infrastructure supporting external API

- Description updated

#9 - 08/09/2018 01:09 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~Ic2aa8137134e92789d22ca86c1daf3c73e7a0297
Gerrit URL: https://gerrit.gromacs.org/8158

#10 - 08/09/2018 01:11 PM - Eric Irrgang

When change 8158 is merged, the roadmap list and chart can be removed from this Redmine Issue in favor of the versioned document in the repository.

#11 - 08/09/2018 01:15 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I1eb8ea75fdaea77e0ce03f2d312c44db8df16f28
Gerrit URL: https://gerrit.gromacs.org/8141

#12 - 08/09/2018 02:33 PM - Joe Jordan

One question comes to mind reading the list. In 29 are we sure we don't want to write tpr files? There are many analysis tasks that require info from to tpr file and not having one that corresponds to a simulation has occasionally caused me some headaches. Would writing tpr files be more acceptable if they were e.g. in JSON and wrote faster than they currently do? If there is an idea that the api will be able to set mdp options on the fly, there will at least need to be some log of which parameters correspond to which simulations.

#13 - 08/09/2018 11:30 PM - Peter Kasson

The thought is that TPR file-writing should be permitted but not required. There are obvious API tasks where writing one TPR per simulation run would be prohibitive.   The plan (at least at the top level) is to log changes, but a complete TPR is not feasible.
Consider the following:
1000 simulation replicas doing 2D replica exchange along temperature and some lambda parameter. Rather than 1000 near-identical TPR files (particularly if the structure itself is large), it is much more efficient for any variances from the TPR to be logged.

#14 - 08/14/2018 03:32 PM - Eric Irrgang

Joe Jordan
I'm not sure if you are saying the headaches are from not having the TPR available while wanting to use the tool, that you have TPR files that are improperly detected to be associated with the data to be analyzed, or something else, but my observation is that many tools require a TPR file, though they only need a subset of the types of information in a TPR file, so API access to such tools should be able to provide just this subset of data and not require a TPR file.

The high-level workflow management tools express data dependencies, so it is the framework's responsibility to make sure that analysis is being performed on the results of a specific simulation. In the simplest case, this can involve TPR files, but we would also like to be able to apply analysis tools to data that is "in flight". Item 29 is more oriented towards rapid and low-cost programmatic variation of inputs, such as for arrays of simulation parameters, structure fitting, etc. I think generating on-disk artifacts are important in many cases, but should be separate API operations from operations used in pipelines or high-throughput situations. Also, a standalone tool like `grompp` should be modular, with distinct parts that generate or transform data versus writing it out in any particular file format. Item 29 is a precursor to rewriting `grompp` in terms of API calls for file reading, data manipulation, and file writing, ensuring that other API clients have the same behavior and compatibility.

In addition to @Peter 's comments,

1. It is appropriate for an API client to be assured that additional metadata is being written and checked, but it should not be required to be a part of the TPR file, or even on the filesystem.
2. The infrastructure to support a workflow-level API is going in a direction that requires operations like changing an input parameter to be part of the checkpoint data for a phase of work, so in addition to being logged in human-readable form, it will be rigorously tracked by the workflow API.

#15 - 08/14/2018 04:48 PM - Eric Irrgang

With regard to gmxapi milestone 16 (exception firewall), @Roland asks "What's wrong with throwing exceptions out of the API?"

to which @Peter responds

> The high-level goal is that in a series of API operations, potentially running in parallel, one exception should cause a failure state for that API operation but not necessarily terminate the operations that might be running independently in parallel. Obviously if operation A triggers and exception and operation B depends on operation A's output, operation B will not run, and all this will be logged. But in terms of overall design, the exceptions shouldn't propagate in a way that they terminate the whole "workflow". Where that exception handling should happen is an interesting design question; my understanding of this from discussing with Eric is that the individual API operation's state would reflect the exception, and I think that's what he is referring to here.

To expand, only the most basic clients can conceivably catch exceptions thrown by the API and do something useful with them. There are many higher-level use cases in which catching an exception is no possible or practical.

gmxapi milestone 16 is about establishing an API boundary across which exceptions will not be thrown. This will allow more unified handling of operation failures and make it easier to build high-level API features without access to libgromacs internals or even the process performing the operation. gmxapi milestone 16 is blocked by milestone 17 to provide API resources for tracking operation results. I have created Redmine issue #2610 to provide a home for that discussion.

#16 - 08/19/2018 07:29 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I40d9931d24560d22b2df72e6cb58db9a60c32f10
Gerrit URL: https://gerrit.gromacs.org/8143

#17 - 08/22/2018 11:15 AM - Joe Jordan

My thoughts on this roadmap are by no means the final word on this, but I foresee a number of problems. The main one is that it seems unlikely that the framework necessary to ship even a skeleton of the gmxapi with gromacs-2019 will be ready in time. This leads to my second large concern. I think it is fine to have 'experimental' features ship with the main code as long as they have an actual use case. This is a major change to how the code works and so is in a sense its own use case, but until it adds some kind of 'feature' experimental or otherwise that a user could play with, it seems like this might want to live in a branch of some kind. Another issue with the roadmap is that many of the milestones are part of the ongoing development and modernization efforts of gromacs. It seems like those efforts should live on master while other aspects of gmxapi should live on a branch. I have a sense for which milestones are in which category, but it would be good to see this written out explicitly.

#18 - 08/22/2018 11:49 AM - Paul Bauer

Adding my two cents to the discussion here, mainly from a developer documentation point of view for change https://gerrit.gromacs.org/#/c/8158/
At the moment I'm not sure how external people would benefit from the information that is provided in the new documentation, besides the point of knowing that the API is under active development.

I think having the roadmap in the main repository has some advantages, but instead of just showing the different milestones it would also need to make clear what is currently possible at the stage of implementation. Otherwise people that want to work with or develop on top of the API might get confused about the current abilities that are already merged (as for now there is no indication which milestones have been reached already).

I would be more than happy to see that we can get this included with subpages for passed milestones.

#19 - 08/22/2018 12:14 PM - Peter Kasson

FYI, the promise is that all of the gerrit changes to implement these milestones will be posted no later than Aug 31. Feedback on specific design considerations is very much appreciated (and requested). Saying "it won't be ready" is, well, unhelpful.

If you'd like to see an implemented, working version, please check out the paper: https://doi.org/10.1093/bioinformatics/bty484
and the github repository with both a functional API and a forked version of Gromacs to accompany.
https://github.com/kassonlab/gmxapi

So we do have a branch. But the goal is to have this available outside our fork (otherwise we end up with a userbase for the Python interface that's dependent on a forked version of Gromacs and the codebase becomes fragmented, which isn't a good design scenario).

#20 - 08/22/2018 04:53 PM - Eric Irrgang

Paul Bauer wrote:

> I think having the roadmap in the main repository has some advantages, but instead of just showing the different milestones it would also need to make clear what is currently possible at the stage of implementation. Otherwise people that want to work with or develop on top of the API might get confused about the current abilities that are already merged (as for now there is no indication which milestones have been reached already).
>
> I would be more than happy to see that we can get this included with subpages for passed milestones.

This is my intention. I was reluctant to make my other Gerrit changes dependent on https://gerrit.gromacs.org/8143 without some feedback on whether this was an appropriate framework in which to track project status.

I would be happy to integrate with other roadmap documents. Unless there are other documents in the developer resources in the repo, though, I think it is best to link these milestones against other works in progress through the issue tracking system and use this issue (and associated documentation) to track the state of meeting functional requirements. If some milestones are trivial due to parallel efforts, that would be great! If you notice it before I do, please let me know! :-)

#21 - 08/22/2018 05:03 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~Ibb16d1453003213a49622810ed8bad4ed4b06e2d
Gerrit URL: https://gerrit.gromacs.org/8219

#22 - 08/25/2018 10:51 PM - Gerrit Code Review Bot

Gerrit received a related DRAFT patchset '2' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I0eeb1d15120d3eb5309ec38c57e4e6f60097885a

Gerrit URL: https://gerrit.gromacs.org/8233

#23 - 08/25/2018 10:51 PM - Gerrit Code Review Bot

Gerrit received a related DRAFT patchset '2' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I23675b4f3bd022ea6986b2e95ca838185327a8b7
Gerrit URL: https://gerrit.gromacs.org/8235

#24 - 08/26/2018 04:03 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~Ib63b1b1a56b04cca145e44d7f08bd011f2150913
Gerrit URL: https://gerrit.gromacs.org/8254

#25 - 08/26/2018 04:20 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~If3b5f63accaffdb3b5963c3098237278650aea92
Gerrit URL: https://gerrit.gromacs.org/8255

#26 - 08/27/2018 12:43 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I206adb7341b3005dd56137e7c7df207a08dd41c2
Gerrit URL: https://gerrit.gromacs.org/8244

#27 - 08/27/2018 04:01 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I542b941c9f2e620934fd2efbd6f71b7f5ce67a63
Gerrit URL: https://gerrit.gromacs.org/8256

#28 - 08/27/2018 05:34 PM - Gerrit Code Review Bot

Gerrit received a related patchset '4' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~Iee677de6f8740b103651c2cb6a0bcdddcf9b0b02
Gerrit URL: https://gerrit.gromacs.org/8215

#29 - 08/27/2018 08:22 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~Ie57962a739ae722b1ced8dd75f0037f07be3e1fa
Gerrit URL: https://gerrit.gromacs.org/8241

#30 - 08/28/2018 07:10 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I2c9bb99d88a11893201489757453ef25bc35a52
Gerrit URL: https://gerrit.gromacs.org/8242

#31 - 08/28/2018 07:10 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I41fb13646c41b0f6e7690a0cedabd17eb3c47433
Gerrit URL: https://gerrit.gromacs.org/8245

#32 - 08/28/2018 07:10 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I48d57570721223eacca957c9a0f7ae98163ab6ff
Gerrit URL: https://gerrit.gromacs.org/8257

#33 - 08/28/2018 07:10 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2585.

Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I61f7b52f1490e7692cdf7f14c7f46f8970b1c30a
Gerrit URL: https://gerrit.gromacs.org/8246

#34 - 09/01/2018 04:34 PM - Eric Irrgang

- Description updated

#35 - 09/01/2018 04:51 PM - Eric Irrgang

- Description updated

#36 - 09/07/2018 01:41 PM - Eric Irrgang

- File Screen Shot 2018-08-26 at 4.19.13 PM.png added

With rebasing and such, the instantaneous dependency relationship of changes in Gerrit is hard to see, so I am attaching a graphic from early in the process. The most blocking changes have been parallelized as much as possible, but this helps illustrate which changes are in support of which. I will add "topics" to the Gerrit changes to help distinguish between pure GROMACS infrastructure to support the API implementation (gmxapi_infrastructure) and changes that add to the external API (gmxapi). I will also start updating the gmxapi feature changes to clearly claim support of a given feature that client code can start looking for immediately (before the next gmxapi spec is finalized).

I appreciate feedback on documentation on structure, but please clarify what changes are blocking inclusion of https://gerrit.gromacs.org/c/8141/ versus incremental changes that can be developed in parallel.

#37 - 09/13/2018 04:23 PM - Eric Irrgang

- File deleted (Screen Shot 2018-08-26 at 4.19.13 PM.png)

#38 - 09/13/2018 04:24 PM - Eric Irrgang

- File gmxapi commit dependencies.png added

#39 - 09/26/2018 05:11 PM - Eric Irrgang

- File gmxapi commit dependencies.png added

- File deleted (gmxapi commit dependencies.png)

#40 - 10/12/2018 08:45 AM - Mark Abraham

We need also to make the include sorting and related module level declarations work smoothly (as well as remove the public api declarations from libgromacs). See discussion at https://gerrit.gromacs.org/#/c/8480/

#41 - 10/12/2018 09:09 AM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2585.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~master~Ib84a89bfa7d5eb54a5a3585d65725fedf98fbab4
Gerrit URL: https://gerrit.gromacs.org/8480

#42 - 10/14/2018 09:12 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2585.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~master~I9909e26a41fd2015667c2f86335c09b6f9708e99
Gerrit URL: https://gerrit.gromacs.org/8550

#43 - 10/15/2018 09:03 AM - Eric Irrgang

- File deleted (gmxapi commit dependencies.png)

#44 - 10/15/2018 09:31 AM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2585.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I3da0d130b07862abc90387ade781d111ccbcd740
Gerrit URL: https://gerrit.gromacs.org/8549

#45 - 10/15/2018 04:19 PM - Eric Irrgang

- File chart.svg added

- File chart.gv added

- Description updated

gmxapi milestones chart has been updated to reflect completed milestones (green) and in-progress milestones (blue). This is the state of completion expected for the GROMACS 2019 release.

#46 - 10/15/2018 04:20 PM - Eric Irrgang

- File deleted (chart.gv)

#47 - 10/15/2018 04:21 PM - Eric Irrgang

- File deleted (chart.svg)

#48 - 11/14/2018 09:12 AM - Eric Irrgang

- Tracker changed from Feature to Task

#49 - 11/14/2018 09:12 AM - Eric Irrgang

- Tracker changed from Task to Feature

#50 - 11/14/2018 09:14 AM - Eric Irrgang

- Description updated

#51 - 03/02/2019 01:35 AM - Eric Irrgang

- Status changed from New to Resolved

This tracked issue has lost its utility as a project management document. The described development flow is no longer relevant for the incomplete tasks. Design and development for the 2020 release will be tracked under #2045 or some other task management structure.

Files

chart.svg	37.6 KB	10/15/2018	Eric Irrgang
chart.gv	4.12 KB	10/15/2018	Eric Irrgang