GROMACS - Feature #2605

Feature # 2585 (Resolved): Infrastructure supporting external API

Library access to MD runner

08/09/2018 02:26 PM - Eric Irrgang

Status:	Closed
Priority:	Normal
Assignee:	
Category:	gmxapi
Target version:	
Difficulty:	uncategorized

Description

gmxapi milestone 4 as described in #2585

Separate user interface from Mdrunner initialization. mdrun CLI program becomes an API client. Creation of an Mdrunner is concise and local. Clients do not handle an Mdrunner that is not initialized (ready to run). Clients can compose the runner with a builder that is independent of the form of user interface.

Relates to #2229

Related issues:

Related to GROMACS - Feature #2229: Full Object Oriented Modularization of GR...	**New**
Blocks GROMACS - Task #2375: Clarify execution phases for MD simulation	**New**
Blocks GROMACS - Feature #2587: Provide Context (e.g. to runner code) to mana...	**In Progress**

Associated revisions

Revision 01ce0f82 - 09/14/2018 02:38 PM - Eric Irrgang

Rearrange mdrun/runner.h

Swap the public and private sections of Mdrunner. Improves readability and reduces noise / conflicts for other changes under review. This commit is only reorganization.

Refs #2605

Change-Id: Ibb9a24a588afd90c7199f77e80aa2a3f2d2e7339

Revision f3ff9147 - 10/03/2018 01:33 PM - Eric Irrgang

Move mdrun mainFunction to client code.

supports gmxapi milestone 4, described at #2605

API clients need to be able to initialize and run MD simulations with calls from separate code blocks or even translation units. Also, we need clear distinctions between a pre-launch master Mdrunner and the non-master Mdrunner threads.

In this change:

- Insert a Builder on which to develop the distinction between user interface and implementation code. User interface is handled in Director code, while the Builder and code further down the mdrun call stack should be user interface agnostic (i.e. not CLI-centric).
- Introduces a Context object to hold some resources. (see milestone 5 and issue #2587)
- Prepare for more stateful Mdrunner objects, documenting resources in need of clearer management, describing future use cases and possible implementation details.
- Make Mdrunner non-copyable and not constructable by clients. Worker threads can initialize a new Mdrunner from the master instance with cloneOnSpawnedThread.
- Clarify parameter setting for spawned Mdrunner threads.
- Begin to encapsulate filename options available to client code from

the actual client code.
- Remove extraneous variables `nfile` and `fnm`.

Later changes will:

- Separate Mdrunner into Launcher and Worker at mdrunner() call so that tMPI runner does not look reentrant.
- Hide data and clarify ownership / modernize memory management / clarify the subjects of the Builder `add` operations.
- Clarify separation of user interface from API parameters.
- Flesh out SimulationContext and related classes.

Refs #2605

Change-Id: I1db1d34b07ec0f8ba5f246ab763c74ad9eafe8f3

Revision 2cf2343a - 10/05/2018 01:04 PM - Eric Irrgang

Add gmxapi::System.

Allow a client to hold an abstract representation of a simulation system
described by a TPR input file. This is part of a series of changes to
allow MD simulations to be configured and launched through a high-level
API.

Supports gmxapi milestone 4

Refs #2605

Change-Id: I6a292a3fe91556d06b9c384b1480c891decffd3f

Revision e39949c7 - 10/01/2019 07:24 AM - Mark Abraham

Removed dependency on commrec of mdrun setup

Changes no functionality.

Setup is now parameterized directly on MPI_COMM_WORLD, which we will
want later for letting library-based callers pass in an
MPI_Communicator. This permits commrec to be initialized later, once
the threads have been spawned for the thread-MPI ranks.

The initialization of multi-simulations moves from LegacyMdrunOptions
to SimulationContext, which is more appropriate for
ensemble-parallelism established directly by the user.

Before the decision about the duty of a rank, there is no difference
between MASTER and SIMMASTER, so several calls to macros
taking a t_commrec pointer are replaced by booleans. Introduced
findIsSimulationMasterRank to compute that value. This eliminates
early use of t_commrec that has necessitated other hacks and
workarounds.

Removed redundant check for replica exchange when the number of multi
simulations is less than two, because gmx_multisim_t constructor
already prohibits that.

Resolves several TODO items and improves modularity, too.

Refs #2587, #2605, #3081

Change-Id: I48bd3b713bc181b5c1e4cbcd648706a9f00eab96

History

#1 - 08/09/2018 02:29 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2605.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~Id304d108792be0bedeafe74d930e4993475959e8
Gerrit URL: https://gerrit.gromacs.org/8152

#2 - 08/09/2018 02:32 PM - Eric Irrgang

- Related to Task #2375: Clarify execution phases for MD simulation added

#3 - 08/09/2018 02:32 PM - Eric Irrgang

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

#4 - 08/09/2018 02:33 PM - Eric Irrgang

- Related to deleted (Task #2375: Clarify execution phases for MD simulation)

#5 - 08/09/2018 02:33 PM - Eric Irrgang

- Blocks Task #2375: Clarify execution phases for MD simulation added

#6 - 08/09/2018 02:34 PM - Eric Irrgang

- Blocks Feature #2587: Provide Context (e.g. to runner code) to manage client and runtime environment added

#7 - 08/21/2018 08:01 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2605.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I1db1d34b07ec0f8ba5f246ab763c74ad9eafe8f3
Gerrit URL: https://gerrit.gromacs.org/8213

#8 - 08/27/2018 01:37 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2605.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~I6a292a3fe91556d06b9c384b1480c891decffd3f
Gerrit URL: https://gerrit.gromacs.org/8239

#9 - 09/07/2018 03:19 PM - Eric Irrgang

https://gerrit.gromacs.org/8213 attempts to separate the handling of user input from the initialization of the gmx::Mdrunner. Since a lot of the member data of Mdrunner is related to potentially optional components, a Builder pattern seemed appropriate. Since the ultimate modularization of the components is not yet clear, the builder interface probably gives people pause.

Depending on whether the Mdrunner should own its members or not, one alternative would be to capture all of the bits and pieces required for the simulation into some other object, tasked with dispensing resources. Then the new runner would just be created with a handle to the SimulationContext passed in, and the runner can interact to get what it needs with a protocol that is not exposed to client code. This would also be consistent with #2587 and https://gerrit.gromacs.org/8215 and other long term plans.

It is also possible that the builder is the right solution, but that we can design a better interface right now.

Current members:

```
        //! Parallelism-related user options.
        gmx_hw_opt_t             hw_opt;
        //! Filenames and properties from command-line argument values.
        std::array<t_filenm, nfile> filenames;
        //! Output context for writing text files
        gmx_output_env_t                       *oenv = nullptr;
        //! Ongoing collection of mdrun options
        MdrunOptions                            mdrunOptions;
        //! Options for the domain decomposition.
        DomdecOptions                           domdecOptions;
        //! Target short-range interations for "cpu", "gpu", or "auto". Default is "auto".
        const char                             *nbpu_opt = nullptr;
        //! Target long-range interactions for "cpu", "gpu", or "auto". Default is "auto".
        const char                             *pme_opt = nullptr;
        //! Target long-range interactions FFT/solve stages for "cpu", "gpu", or "auto". Default is "auto".
        const char                             *pme_fft_opt = nullptr;
        //! Command-line override for the duration of a neighbor list with the Verlet scheme.
        int                                     nstlist_cmdline = 0;
        //! Parameters for replica-exchange simulations.
        ReplicaExchangeParameters               replExParams;
        //! Print a warning if any force is larger than this (in kJ/mol nm).
        real                                    pforce = -1;
        //! Handle to file used for logging.
        FILE                                   *fplog {nullptr};
        //! Handle to communication data structure.
        t_commrec                              *cr {nullptr};
        //! Handle to multi-simulation handler.
        gmx_multisim_t                         *ms {nullptr};
```

I would propose that several of these are not *optional* but should be considered part of an abstraction of the execution environment; in the long run,

an extension of ProgramContext. With that in mind, I propose splitting up responsibility for these parameters as follows.

- hw_opt: Context
- filenames: Context (until it can be absorbed by other modules)
- oenv: Context
- mdrunOptions: SimulationMethod (most, but not all, of these are run-time parameters that should not affect simulation results, though they could be absorbed into runtime parameters for more specific modules)
- domdecOptions: DomainDecomposition
- nbpu_opt: NonBonded
- pme_opt: Electrostatics
- pme_fft_opt: Electrostatics
- nstlist_cmdline: NeighborList
- replExParams: ReplicaExchange
- pforce: SimulationMethod
- fplog: Context
- cr: Context
- ms: MultiSim

If implemented, client code would create and launch a Mdrunner similarly to the following.

```
auto context = gmx::SimulationContext(cr, hw_opt, filenames, oenv, fplog);
auto builder = gmx::Mdrunner::Builder(context);
builder.addSimulationMethod(mdrunOptions, pforce);
builder.addDomainDecompositon(domdecOptions);
builder.addNonBonded(nbpu_opt);
builder.addElectrostatics(pme_opt, pme_fft_opt);
builder.addNeighborList(nstlist_cmdline);
builder.addReplicaExchange(replExParams);
builder.addMultiSim(ms);
auto runner = builder.build();
try
{
    runner.run();
}
catch (const GromacsException& e)
{
    // handle simulation failure scenarios...
}
```

It's a little weird not to require simulation method, so that could be moved to the builder constructor as well.

#10 - 09/14/2018 12:45 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2605.
Uploader: M. Eric Irrgang (ericirrgang@gmail.com)
Change-Id: gromacs~master~Ibb9a24a588afd90c7199f77e80aa2a3f2d2e7339
Gerrit URL: https://gerrit.gromacs.org/8352

#11 - 10/15/2018 03:29 PM - Eric Irrgang

- Status changed from New to Resolved

In addition to establishing Mdrunner creation through a builder, the legacy user options comprising most of the mdrun program have been separated out into a separate (monolithic) structure that we can chip away at during the 2020 development cycle.

#12 - 01/07/2019 01:39 AM - Mark Abraham

- Category set to gmxapi

- Status changed from Resolved to Closed