GROMACS planning meeting

18 January 2023, 13:00 - 17:00 CET
Zoom link

Tentative time-planning / things to discuss:

13.15 API discussions (+python?)

- Stockholm work on core data structures / API - prio during 2023, but long-term effort
- Need for cleanly specified parameters/settings
- Clear rules for when parameters can change and how notifications happen across modules.
- Need for clean & stable ForceProvider interface. Thoughts that the current way parameters are specified is a bit dirty/inflexible. Not well expressed in C++ code
- How do mdmodules relate to core datatypes/API

14.00 Prerequisites/standards/choices for release-2024

- Stick to C++17 one more year, unless interfaces are already stable across the C++ compilers we need (unlikely).
- Maintain CUDA-11
- Require the latest SYCL version released during 2023 for gromacs release 2024
- Upgrade to require the latest cmake release today (which will be second-latest or so by release-2024) if there are specific features that would save significant developer time
- Challenge with diversity of GPU FFT libraries. Try to consolidate, or at least document to users what we invest our main effort in, and be clear the others are more intended for testing.

14.30 GPU/acceleration/parallelization

- Lots of work on CUDA graphs
- Interest (among HPC centers) to optimize for energy use rather than performance. Change clocks even during runs. Alan gives talk at GTC.
-

15.00 Coffee break
15.15 Tools, ensemble simulations, collective variables, cpH, ForceProvider, etc.
15.45 Training, workshops, forums, tutorials, user-driven development
16.15 Tentative scheduling of effort/people
16.45 Next steps, wrap-up planning subgroup meetings

Erik Lindahl:

- Increase max # atoms to 2^63. Not important short-term for us, but strategically extremely valuable in order to position ourselves for whole-cell modeling where 2 billion particles is insufficient - and this will likely be an area of great interest to SciLifeLab (read: local long-term funding).
- Implement RISC-V acceleration (funded by EU PILOT, strategically important)
- Time to work on file formats both for input & output (and sharing data efficiently) - funded as part of the MDDB project, we will be recruiting 1-2 postdocs for it
- BioExcel-3 starting (after 6 months of break after BioExcel-2) means an opportunity to start addressing a bunch of things that are important but not urgent, since we don't have any really tough deliverables next 6-12 months, and need to recruit:
    - BioExcel-3 will mean stricter requirements that development should be user-driven/prioritized. During the year we'll develop some sort of questionnaire or form for this. We will still have a lot of power to influence things, but need to get used to first explaining, asking users from priorities, and then doing work
    - BioExcel-3 will be more focused on user experience than performance/scaling, but this is also an opportunity to revisit a lot of old sins and provide better tools for system building & simulation setup. Recruitment ads in progress.
    - Improve user interfaces for working with ensemble simulations - key usability target for BioExcel, funding available.
    - Tutorials & training (again part of BioExcel)
    - Interfacing GROMACS with NeuralNetwork-derived potentials, first as proof-of-concept, second stage as delta-learning where we evaluate both Lifson & AI forces each step. Funded as part of AQTIVATE MarieCurie graduate school. We should be able to combine this with an overhaul of a more isolated module for nonbonded interactions, which will also provide the effort needed for tabulated nonbonded kernels and a lot of other nonbonded features.
- We will hire a new project manager as part of BioExcel-3, but this will be more focused on actual project management, not having a central person doing a lot of code review and branch merging. This will have to be more distributed among developers who want their own code contributions to be reviewed :-)
- A lot of free energy work funded as part of LIGATE (see Sebastian's comment)
- A lot of API work foreseen (where we plan to have new people/effort from a number of the projects above), but likely not until Q2/Q3, partly because we need to recruit. I would prefer if we start by cleaning up some old "sins" so we avoid hardcoding concepts like MDP/TPR and a necessary "grompp-like" step ahead of running simulations in the API.
- More work on exhaustive unit testing would radically shorten our release phases

Sebastian Wingbermühle:

- Complete plans for more versatile free energy/enhanced sampling in GROMACS:
    - https://gitlab.com/gromacs/gromacs/-/issues/4014
    - https://gitlab.com/gromacs/gromacs/-/issues/4431

    - https://gitlab.com/gromacs/gromacs/-/issues/4432
- This dev-cycle:
    - Focus on FEP/REST2 support (second half of 2023)
- Large-scale testing of RBFE with AWH
- Automated workflow for RBFE calculations with GROMACS

Joe Jordan

(note I won't be able to attend due to a project kick-off meeting)
Demo nblib listed forces calculator in context of a real/full MD step. This is still a prio for CSCS so hopefully this will be the year.

Berk Hess

- Constant-pH code integration https://gitlab.com/gromacs/gromacs/-/issues/4273
- Tabulated non-bonded interactions https://gitlab.com/gromacs/gromacs/-/issues/1347
- We should discuss strategies for APIs

M. Eric Irrgang:

If any of the following outstanding issues are of interest, please advise. Otherwise, I will close them soon with some final notes.
Other than that,
The main thing I would propose to plan for is a unified effort to simplify (flatten) the mdrun call stack: strip out MdrunnerBuilder, Mdrunner, and Mdrunner::mdrunner, and expose SimulatorBuilder. (If encapsulating the participating module details is not possible, the intermediate logic can at least be encapsulated in Director functions that act on the SimulatorBuilder, or factory functions/ParameterTypes for the SimulatorBuilder and MDModules inputs.) If there is sufficient interest in making such a plan, @eirrgang could investigate additional short-term funding for tightly-scoped contributions.
ericirrgang@gmail.com

packaging and build system

- Help clients find MPI-enabled GROMACS
- Client software build support. (#4447) · Issues - GROMACS
- `pip install gmxapi` needs better handling when multiple GROMACS APIs are available. (#4335) · Issues
- CMake `gmxapi` package should account for multiple GROMACS flavors installed to the same prefix.

- I can also look again at building the gmxapi Python package against a Conda gromacs installation so that we can port the gmxapi tutorial material.

gmxapi source improvements

- Remove `gmxapi::System`
- Simplify Python metaprogramming in gmxapi package.

gmxapi maintenance

- Compatibility cross-check for gmxapi Python package and older GROMACS releases

gromacs library and API

structure and infrastructure

- MPI_COMM_WORLD remediation
- Clean up library context management. (#3688) - GROMACS
- Use build system infrastructure instead of custom scripts to manage API levels. (#3288)
- Infrastructure and patterns for expressing public interfaces

interoperability and client support

- Clarify distinction between public API documentation and developer docs.
- Scalar and structured type expression and definitions for API

features

- Public API for ForceProviders (this really warrants a simplification of the MD launch, including a flattening of the call stack and exposure of the SimulatorBuilder, but there has not yet been sufficient interest to result in a tracked issue)
- Update gmxapi simulation setup protocol.
- Refine SimulationInput (#3652) · Issues - GROMACS
- C++ API for simulation input and output
- Expand gmxapi.modify_input use cases.

Mark Abraham:

No clear plans for Intel-related feature work this quarter. Intel needs to do some profiling and prototyping before we will have proposals. I'm interested in several things on general principles, like

- Resolve SYCL- and/or PVC-related issues identified for main during release-2023 dev cycle

- Continuing feature completion of modular simulator (Pascal's shared his dev branch on gitlab) so we can consider deprecating legacy integrators and/or implementing GPU integrators there
- Continuing to improve mdrun startup so that checkpoint reading happens ASAP after tpr reading, to simplify code and suit API use cases
- Does PVC really need a work-around for lack of hardware support for high priority queues?
- Explore how to implement GPU-based pair search
- Rework PME spread to gather particles to grid points, avoiding needing atomic writes
- Using an approach like (or the approach of) the modular simulator to build force-calculation schedules (perhaps targeting a prototype SYCL extension for graph execution)

but I do not foresee proposing significant code changes this quarter.

Magnus Lundborg:

Features

- Optimized AWH target distribution based on AWH friction metric (#3843).
    - Working branch ml_awh_friction_optimized_target_distribution_2023.
    - In the process of verifying that there is any improvement when optimizing the target distribution.
- SD integrator updates and constraints on GPU (#4658).
    - Working branch 2023_sd-update-gpu_awh-symm_awh-corrblocks.
    - Only for CUDA so far.
    - Should be split into separate MRs:
        - Random number generation - needs tests
        - Adding updates and constraints.

Peter Kasson:

Will be able to join for end of meeting
ForceProvider API (we have some applications that need a slight expansion of current capabilities; it would be a great time to consider the general API, or we could go for the minimum required feature set).

Colvars devs: Hubert Santuz, Jérôme Hénin, Giacomo Fiorin

Working on feature-complete MdModules version
(see this branch: https://gitlab.com/HubLot/gromacs/-/tree/colvars-interface)
Checkpointing is making progress
Still need an ensemble temperature notifier as agreed previously

- Bulk of the Colvars code lives in external/

- Master branch of the library is validated by CI regression tests when built against several Gromacs releases and other codes.

Szilárd Páll:

- SYCL: [1]
  - Portability: broader implementation support
  - Performance: improve hipSYCL/AMD GPU performance, improve kernels (bonded)
- PME decomposition follow-up #3884 and related [2]
- DD / PP parallelization: GPU DLB, NVSHMEM, consider kernel fusion [3]
- Continue improving the md loop / schedule modularity (more use of workload flags, refactor/move search/DD) [4]
- Graph scheduling: enable CPU tasks and MPI communication (related to [4]) [5]

Alan Gray (NVIDIA):

- Strong overlap with Szilárd (above).
- CUDA Graph enhancements - extend to support CPU forces and libMPI/NVSHMEM
- NVSHMEM GPU Communications, as a faster alternative to MPI on multi-node
- Single-node multi-GPU enhancements with thread-MPI
- Energy/Power Optimization

Andrey Alekseenko:

- SYCL:
  - Further testing, usability improvements
    - Code unification/cleanup
    - Device context rework, FFT and GPU-aware MPI handling, device buffer improvements, …
  - Runtime performance investigation / improvements
- MD loop / schedule improvements.
  - Existing scheduling frameworks?

Alessandra Villa

I will go on to take care of web-page, forum community, GROMACS training and tutorials.

More specific for this year: design of developer GROMACS school, quick tutorials on GROMACS tools.

I plan to work on some small tool refinement and on AWH free energy.

Cathrine Bergh

- Continue the implementation of gmx msm
- We need someone to take over the work Paul was doing on gmx cluster and the TAF (e.g. we need support for multiple trajectories)?