GROMACS - Feature #868

Implement parallelization support to analysis framework

01/26/2012 07:13 PM - Teemu Murtola

Status:	In Progress	
Priority:	Normal	
Assignee:	Kevin Boyd	
Category:	analysis tools	
Target version:	2021	
Difficulty:	uncategorized	

Description

The trajectory analysis framework in master has been designed such that it should be possible to implement parallelization relatively easily such that different threads/processes would process independent frames. MPI and thread parallelization may require partially different approaches here for good efficiency, but some refactoring effort is likely common for both.

The main things that need to be considered:

- Implement a parallel version of TrajectoryAnalysisCommandLineRunner. I don't have any ready design for this; alternatives include completely separate classes, separate Impl classes, or just conditionals within one single class. Division between the runner and TrajectoryAnalysisRunnerCommon probably needs to be re-evaluated in this context.
- Implement parallel support in AnalysisDataStorage, with possible supporting changes in other analysis data classes. Thread-parallelization should be possible by adding appropriate synchronization primitives in correct places in AnalysisDataStorage, but MPI may require more thought.
- If processing of frames requires information from earlier frames, a simple parallelization implementation may not cope correctly with the situation. It may be necessary to add information to TrajectoryAnalysisSettings to inform the runner that the tool expects this to allow meaningful error messages.

Related issues:

Related to GROMACS - Task #869: Make analysis data histogramming and multipoi...	**In Progress**	
Related to GROMACS - Task #948: C++ thread synchronization primitives	**New**	
Blocked by GROMACS - Task #996: C++ MPI Framework	**New**	**08/30/2012**

History

#1 - 02/17/2012 12:50 PM - Roland Schulz

- Assignee set to Roland Schulz

#2 - 12/30/2012 06:43 AM - Teemu Murtola

- Status changed from New to In Progress

- Target version set to 5.0

At least some level of parallelism is planned for 5.0. Roland and others are working on the MPI parallelization part.

#3 - 03/25/2013 06:24 PM - Mark Abraham

I've taken a look at https://gerrit.gromacs.org/1316 and so far like what I see.

I gather the main focus of development has been to support "embarrassing parallelism," i.e. that frames are read in somehow, and processed typically on only one core. That suits analysis tasks that are relatively compute-intensive and only need one frame. We'd have to review if/when we identified other needs.

Currently, there's some wasted reading going on because the old trajectory-reading framework doesn't support anything but serial access. Ideally, the new TrajectoryReader framework would support

- passing the next frame number selected by the TrajectoryAnalysisScheduler to the TrajectoryReadManager and having the implementation return only that frame (and transparently do that as efficiently as the file format and I/O strategy permits)
- passing some other kind of condition that could be potentially satisfied by reading a trajectory frame header (e.g. "n == t % dt" where n might vary across the set of processors and/or dt vary with the size of the processor set; this would require another implementation of TrajectoryAnalysisScheduler? and would need machinery to cope with the situation where no frame satisfies the condition, without having to read the whole trajectory)
- handling exceptions that can't be fully handled by the TrajectoryReadManager (the latter can add some context information, but mostly it has to bump error conditions up to its client to decide how gracefully it wants to cope)

So far the design in #1193 does those things, though there are details to be decided for how to construct and pass the objects that specify conditions. Anything I've missed?

#4 - 03/25/2013 07:28 PM - Teemu Murtola

Mark Abraham wrote:

> I gather the main focus of development has been to support "embarrassing parallelism," i.e. that frames are read in somehow, and processed typically on only one core. That suits analysis tasks that are relatively compute-intensive and only need one frame. We'd have to review if/when we identified other needs.

Yes, this has been the main focus. Some thought has been given to the interface possibly supporting other types of parallelism (e.g., splitting the atoms to different cores), but not much, and there is nothing concrete on this front. Any work that is not within a frame can be implemented in two ways in the current framework using the features in the analysisdata module: either by 1) storing data from previous frames in an AnalysisData object and accessing that during the analysis (parallel support for this not implemented yet, and may limit parallelizability quite significantly), or by 2) writing that work as an AnalysisDataModule. The main assumption has been that this work is cheap compared to the I/O and per-frame analysis, but it is conceivable to have some parallelization also within an AnalysisDataModule. But if that very easily requires a communication-intensive transpose operation with MPI (but could be achievable with threads).

#5 - 01/01/2014 07:08 AM - Teemu Murtola

- Project changed from Next-generation analysis tools to GROMACS

- Category set to analysis tools

- Status changed from In Progress to Accepted

- Assignee deleted (Roland Schulz)

- Target version changed from 5.0 to 5.x

Not going to happen for 5.0.

#6 - 07/11/2016 08:28 PM - Mark Abraham

- Target version deleted (5.x)

#7 - 02/02/2019 06:05 PM - Kevin Boyd

- Status changed from Accepted to In Progress

- Assignee set to Kevin Boyd

- Target version set to 2020

I have WIP for simple std::thread based parallelism over frames, based on the doxygen flowchart that Teemu put together. If anyone's interested I can add them to the gerrit draft.

Right now (unless I'm doing something wrong), there appears to be an issue with the selections not being thread-safe. I'm getting data races when one thread is evaluating selections and another is doing pair searches, and I'm not very familiar with the selection code.

#8 - 02/02/2019 07:33 PM - Teemu Murtola

The selection code is indeed not thread safe (and cannot be made such, without a lot of work (which may include completely changing the approach)). This part is indeed missing from the description, but the "trivial" solution is to create multiple copies of the selection collection (one for each thread) by parsing the input multiple times.

#9 - 02/02/2019 07:39 PM - Kevin Boyd

Teemu Murtola wrote:

> The selection code is indeed not thread safe (and cannot be made such, without a lot of work (which may include completely changing the approach)). This part is indeed missing from the description, but the "trivial" solution is to create multiple copies of the selection collection (one for each thread) by parsing the input multiple times.

Thanks, that's super helpful, will do!

#10 - 02/04/2019 06:22 PM - Kevin Boyd

Teemu Murtola wrote:

> The selection code is indeed not thread safe (and cannot be made such, without a lot of work (which may include completely changing the approach)). This part is indeed missing from the description, but the "trivial" solution is to create multiple copies of the selection collection (one for each thread) by parsing the input multiple times.

Would it make sense to make the SelectionCollection copyable instead, and copy the class after everything's been compiled? Parsing the input multiple times doesn't make sense if e.g. it's in interactive mode.

#11 - 02/04/2019 06:48 PM - Teemu Murtola

Kevin Boyd wrote:

> Teemu Murtola wrote:
>
> > The selection code is indeed not thread safe (and cannot be made such, without a lot of work (which may include completely changing the approach)). This part is indeed missing from the description, but the "trivial" solution is to create multiple copies of the selection collection (one for each thread) by parsing the input multiple times.
>
> Would it make sense to make the SelectionCollection copyable instead, and copy the class after everything's been compiled? Parsing the input multiple times doesn't make sense if e.g. it's in interactive mode.

Sure, if someone has time to implement and test that... I estimate this to be at minimum several hundred lines of code, and would need to be tested in many of the cases covered by the current ~200 tests for the selection code. The data structure is a relatively complex hierarchy of elements (essentially, a DAG) with some elements owning memory, and some others pointing to memory owned by others.

#12 - 02/04/2019 07:36 PM - Kevin Boyd

OK in that case it's probably easier to go with your first suggestion. So for each thread, I'll have to manually set up the selections. That's normally run by the options module. I'm a little unclear on how to do it manually - would I create a default SelectionCollection, then setTopology and setIndexGroups? Or is there more to it? Thanks for the help BTW!

#13 - 02/04/2019 07:47 PM - Kevin Boyd

Kevin Boyd wrote:

> OK in that case it's probably easier to go with your first suggestion. So for each thread, I'll have to manually set up the selections. That's normally run by the options module. I'm a little unclear on how to do it manually - would I create a default SelectionCollection, then setTopology and setIndexGroups? Or is there more to it? Thanks for the help BTW!

Err, wait, there would definitely have to be more than that, if a string selection is dynamically evaluated frame by frame.

#14 - 02/04/2019 08:34 PM - Teemu Murtola

I would try to still make this a clone() method on the SelectionCollection. I think the collection already has all of the input text stored internally (in pieces), so it could take care of reparsing it. The header explains which methods need to be called and in which sequence.

In order to make it possible to implement TrajectoryAnalysisModuleData::parallelSelection() robustly, you may also need to add a running index to SelectionData or something similar.

#15 - 02/06/2019 01:27 AM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #868.
Uploader: Kevin Boyd (kevin.boyd@uconn.edu)
Change-Id: gromacs~master~I5cc5ae880fa6643a309d3516f3c1478e14820877
Gerrit URL: https://gerrit.gromacs.org/9092

#16 - 02/07/2019 02:08 AM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #868.
Uploader: Kevin Boyd (kevin.boyd@uconn.edu)
Change-Id: gromacs~master~I312ffca935ef08ea98dfd230367fb5c89ea44934
Gerrit URL: https://gerrit.gromacs.org/9102

#17 - 12/20/2019 11:59 AM - Paul Bauer

- Target version changed from 2020 to 2021