GROMACS - Feature #2901

Declare external Resources in mdp / tpr files.

03/21/2019 05:19 PM - Christian Blau

Status:	New	
Priority:	Normal	
Assignee:	Christian Blau	
Category:	preprocessing (pdb2gmx,grompp)	
Target version:	2020	
Difficulty:	uncategorized	

Description

Unify data representation not present in the .tpr files and include information about this data in pre-processing steps (currently via .mdp-files).
To avoid transform experimental data or very large and complex data input to tpr-format would ultimately turn tpr files to a wrapper file format for all sorts of input data sources we use during simulations.

Currently essential dyanmics uses an external data file that is provided as an command line option to mdrun; in the future, all code that relies on external experimental data for driving simulations will rely on this, e.g. cryo-EM fitting, WAXS/SAXS module, secondary structure driven simulations as well as contact driven md.

Instead of the current design, we would rather like to store handles to the data sources in the tpr file format.

These data sources should be

```
- labelled
 - have a specified type
 - handle to the raw data
 - include a source sanity check, if possible
```

Related to #2282, #2590

Associated revisions

Revision 75a273b0 - 04/05/2019 05:21 PM - Christian Blau

densityfitting - MDModule declaration

Declaring the infrastructure for running molecular dynamics simulations with
addional forces that are derived from densities.

Adds a IForceProvider for density fitting simulation that is set up with
its DensityFittingParameters that are in turn built from DensityFittingOptions.

refs #2282 #2901

Change-Id: I0732a78747582a6e23bba1e141d73c4cda421011

History

#1 - 03/21/2019 06:08 PM - Gerrit Code Review Bot

Gerrit received a related DRAFT patchset '1' for Issue #2901.
Uploader: Christian Blau (cblau@gwdg.de)
Change-Id: gromacs~master~If8bafcc10b2720857b2fceca311cf6ba3ea98a5c
Gerrit URL: https://gerrit.gromacs.org/9344

#2 - 03/23/2019 03:00 PM - Erik Lindahl

Commenting here instead to avoid creating noise in Gerrit!

I thought a little bit more about it over lunch, and this might be an alternative way of handling it:

- IMHO, The most important thing both for input & output data is what the data is, not what method is producing or consuming it. We could for instance imagine a dozen algorithms that use a reference density.
- I would like to avoid having the I/O layer encode and be aware of every single file format.

This made me thing of rather designing the I/O handler as a module where each methods module register their supported input/output at initialization - which in the future could also enable users to add more such tools as dynamically loadable objects at runtime.

At registration, Data could then be handled in layers (not sure if inheritance is good...):

1) a low-level description that just encodes format (and maybe optionally units of data), say a 3D density.

2) A mid-level description that fully characterizes the data (say electron density including units)

3) a specific file format description, together with a routine that will try to autodetect if a provided file is of this type (say PDB files provided with extension ENT instead).

I think this would provide a very nice user experience where each module/function can have a rich desiription of the formats supported, and we can add more formats at compile-time by linking with suitable libraries.

I also think it would make for clean modules that never have to modify the IO handler itself.

Thoughts?

#3 - 03/24/2019 09:08 PM - Eric Irrgang

For the gmxapi stuff, I've been expecting that it will take a while to define named schema and map them to/from serialization schemes / URI types. First, and fundamentally, operations have to express their inputs in terms of simple scalar data, simple structured data, and aggregate structured data that can be used as the basis for simplification through defined schema later. In the mean time, the only thing that makes sense with an input filename is to treat it as a String resource outside of the thing that can read the file.

Coming from the other direction, then, I'm working on ways for compute code to express it's input in simple, standard, template-assisted ways that I hope are convergent with concepts for the Options framework.
For example:
https://github.com/eirrgang/sample_restraint/blob/5ce0483b7662843d8b864cffcde656bdb37b293c/src/pythonmodule/export_plugin.cpp#L55

#4 - 03/26/2019 01:54 PM - Mark Abraham

I changed Erik's comment number 2 from private to public.

#5 - 03/27/2019 04:27 PM - Eric Irrgang

Erik Lindahl wrote:

> Thoughts?

Ah! I think that's basically what I was saying here and in Gerrit, but I guess your comment was private at the time I commented. :-)