GROMACS - Feature #2993

Task # 2045 (New): API design and language bindings

Scalar and structured type expression and definitions for API

06/24/2019 01:25 PM - Eric Irrgang

Status:	New
Priority:	Normal
Assignee:	Eric Irrgang
Category:	gmxapi
Target version:	
Difficulty:	uncategorized

Description

Operation implementations need to be able to express their allowed inputs and available outputs in ways that support easy generation of helper code and user interface code, check-pointing, and efficient data interchange across APIs and language boundaries.

To support compartmentalized optimization in scheduling frameworks, data must be representable as Futures.

To reduce the complexity of interactions in complex or "ensemble" data flow topologies, data shape metadata should allow easy accommodation of non-local dimensions and elements.

Note: the GROMACS library includes several internal frameworks that use some amount of named types for template arguments, serialization tools, etc. Compared to the current issue, the two most relevant pattens in the library now are the KeyValueTree types and the mdspan metadata, but both are in the internal library API.

In resolving this issue, we should seek to define types that can be implemented with zero run-time overhead in terms of library data types, but the essential goal is to produce clean and stable interfaces for the public API.

To evaluate compatibility, consider interaction use cases with APIs such as

- Eigen
- XDR
- numpy
- Python buffer protocol
- C++ mdspan
- HDF5
- JSON

Also reference

- GROMACS KeyValue tree
- GROMACS serializer / deserializer
- GROMACS checkpoint infrastructure

Proposal

Proposed data types and protocols will be submitted through Gerrit and should include design documentation, Python implementation details, and C++ implementation details.

Subtasks:

Task # 3130: Interim handling of gmxapi data references.	**New**
Feature # 3140: Allow explicit input definition for gmxapi.operation function wrapper	**New**
Bug # 3141: gmxapi File placeholders missing from beta release	**New**
Bug # 3150: gmxapi data type annotations are confusing and inadequate	**New**

Associated revisions

Revision 1c5fc1fa - 08/02/2019 06:06 PM - Eric Irrgang

Rearchitect gmxapi.operation

Includes many work-arounds for an incomplete data model,
illustrating what will need to be addressed in gmxapi
self-describing type system, data shaping, and Futures
implementation.

- Move several nested classes to the gmxapi.operation scope.
  Introduced abstractions and refactoring to replace some
  dynamic definitions in the function_wrapper closure with
  composition or more tightly scoped closures. Provide
  cleaner helpers for dynamically defined operation, input,
  and output types.
- Introduce minimal NDArray class and ndarray factory.
- Replace ImmediateResult with a Future of a StaticResource
- Implement Future slicing with Futures of ProxyResource
- Define several Descriptor classes for generic attribute
  accessors in standard interfaces, supporting similar
  style of interaction with resources as in C++ extensions.
- Explicitly type collaborations in the preliminary data flow
  protocols.
- Introduce ensemble data.
- Rename append_list to join_arrays.
- Add a lot of static type hinting and run-time type checking.

Note that the execution dependency in FR2 has been superseded by the
data flow driven dependency in FR3. The syntax supported in FR2 is now
disabled to allow development towards FR4.

Refs #2993
Refs #2994
Refs #2996

Change-Id: I94a63d5801f97eb79962c693b48fa80a7c96c0ec

Revision a507ae2a - 08/22/2019 03:04 PM - Eric Irrgang

C++ code and Python bindings for TPR parameter read/write.

Core functionality to allow TPR files to be rewritten with altered
parameters, ported from https://github.com/kassonlab/gmxapi v0.0.7.4
with modifications to naming and coding style. (Refer to patch history.)

A lot of the C++ code is not intended to be a long term solution,
but demonstrates the use cases that will need to be addressed
as modules become able to self-describe their inputs and outputs.

However, this change is necessary to support near-term data portability
between gmxapi operations for preparing and wrangling chains or
ensembles of simulations.

Refs #2993

Change-Id: I54677c861dfb19c9f34b11d2c30456e6ee5dbe8d

History

#1 - 08/21/2019 02:27 PM - Eric Irrgang

From an out-of-band discussion with Mark:

It is worth noting that gmxapi, its clients, and its collaborators need to be able to adequately describe data in ways that allow for no-copy reference-passing. gmxapi scalar types should include a superset of types in interoperating APIs and file formats, such as ISerializer, XDR, numpy, Eigen, etc, that are likely to occur in current or future GROMACS versions. gmxapi structured types should have enough metadata to be compatible with C++ mdspan, Numpy arrays, etc.

These types allow us to describe objects passed into or out of the API by reference, such as with a Numpy array. This is distinct from describing types of data that crosses the API boundary (such as passing data to or from a Python list, dict, or scalar, or from high-level API operations to library-internal data structures). In these cases, we try to allow for no-copy optimizations, but do not guarantee them, and we remind client coders of potential costs by requiring explicit extraction functions, like Future.result().