GROMACS - Feature #2996

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

gmxapi execution model

06/24/2019 02:16 PM - Eric Irrgang

Status:	In Progress	
Priority:	Normal	
Assignee:	Eric Irrgang	
Category:	gmxapi	
Target version:	2021-infrastructure-stable	
Difficulty:	uncategorized	

Description

This issue documents and coordinates the development of the gmxapi execution model supporting the 2020 function requirements outlined in the roadmap document.

Key aspects include the following:

- Computational work and data transformations are defined as Operations, which are instantiated by client code as nodes of a directed acyclic work graph, connected by data flow edges.
- The factories used to get OperationHandles (work graph nodes) are context-dependent. In the basic use case, execution is deferred and the handle obtained provides a means of accessing Futures for the operation outputs.
- gmxapi is fully extensible, such that end users and third parties can write new operations or execution context implementations with well-defined API tools.

In addition to design documentation, supporting software for these tasks include

- Python base classes, meta classes, decorators / functionals, and dynamic type factories.
- C++ template headers and sample code.

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Note that the execution dependency in FR2 has been superseded by the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 5746c459 - 08/05/2019 01:18 PM - Eric Irrgang

Separate out OperationDirector.

Develop abstractions for implementing operations with data flow.

Refs #2996

Change-Id: Ib0b899ea85a59c58ee33a5e3d8a28b33fb67047a

Revision 9fa02702 - 08/06/2019 04:08 PM - Eric Irrgang

Separate out OperationDetailsBase.

Develop abstractions for implementing operations with data flow.

Refs #2996

Change-Id: Ie893039f2a73647ed254e0f4a68ed4e21de7a500

Revision 88da3c14 - 08/12/2019 09:41 AM - Eric Irrgang

Separate out some abstract base classes.

Introduce SourceResource and AbstractOperation to improve consistency
and static type checking.

Refs #2996

Change-Id: If7a23ca66e21e55cd67a6242fd699507699423c0

Revision 16ef7ec0 - 08/12/2019 05:02 PM - Eric Irrgang

Introduce Context requirement.

Operation handles are acquired with mediation from a
specific Context object, which supports a protocol for
adding nodes to a work graph.

Refs #2994
Refs #2996

Change-Id: I4b110eff23a8326bad21bbf2b1b236522d3ba630

History

#1 - 12/10/2019 01:57 PM - Eric Irrgang

- Status changed from New to In Progress

- Target version changed from 2020 to 2021-infrastructure-stable

This task should include documentation, simplification, and updates, mostly to the Python framework for expressing and managing Contexts and Operation definitions, and will need to be addressed over the course of gmxapi development between 0.1 and 1.0.