There are several places in the code where C++ interfaces have been put up a bit in a haste to get a workable framework, and they could be improved significantly. At least the following would be good to improve:

- **Selection class in** source:src/gromacs/selection/selection.h. Currently, this is a very thin wrapper around the original C structure from the selection library in 4.5. It's not used in that many places in the selection code, so it could be completely rewritten to make the interface more C++-like. Some underlying data structures can be hard to change without copying the contents, but it should be possible to wrap them without too much performance impact.
- **NeighborhoodSearch class in** source:src/gromacs/trajectoryanalysis/nbsearch.h is also a very thin wrapper around original C code in source:src/gromacs/selection/nbsearch.h. It would be best to just convert the C code into C++, improving the interface in the process, and get rid of the wrapper. Would be good to add unit tests at the same time to make sure that the converted code works.
- **SelectionCollection class in** source:src/gromacs/selection/selectioncollection.h has its "private" implementation class as a public member, which is ugly. Shouldn't be too hard to change, as the contents are mostly used in selection compilation and evaluation. The C++ interface could be pushed a bit further into the module by making the compilation and evaluation happen in classes that are friends of SelectionCollection.

At least the first should be done before #665 is started seriously, because most analysis tools will use the Selection class interface extensively.
Selection::_sel is now properly private.

Added a simple SelectionEvaluator class and made it friend of the relevant classes. Also moved some functionality to the Selection class from the compiler.

This is a first step in improving the Selection class and completely removing gmx_ana_selection_t.

Part of issue #827.

Change-Id: i53a370361f897a0e736f58a886dd0e7de3cf71b9

---

Removed old gmx_ana_selection_t.

Moved variables previously held there to member variables of the Selection class. Added private member functions to the Selection class to remove direct member variable access from SelectionCompiler and SelectionEvaluator. Makes it easier to restructure the Selection implementation since access to the member variables is now (mostly) limited to selection.cpp.

Part of issue #827.

Change-Id: i60b302409afc1f9535ddd7423a8c21f3799f72e

---

Use frame objects in AnalysisDataModuleInterface.

Instead of passing raw pointers to the internal data in the public interface, use wrapper objects where the individual values are accessed through accessor methods. This makes it easier and more flexible to change the internal data representation without any code changes in classes implementing this interface (recompilation may be necessary because of inlined functions, though). Similarly, it is easier to add/change information that is passed through the interface without touching existing code.

Also added some additional documentation for functors used in analysis data unit tests.

Part of issue #827.

Change-Id: i08d5b9719c0a14040c3172de39c8620d870382af
Pass frame header also to frameFinished().

Frame header information is now passed also to frameFinished() in AnalysisDataModuleInterface in addition to frameStarted(). Currently, this information is not used anywhere, but it makes the interface more symmetric, and allows frameFinished() to easily determine the frame index of the frame that became ready. The latter will be used in future commits.

Related to issue #827.

Change-Id: l8a3635823e2d9d6422d2a37ab4b8ec6b606589a0

Improve AbstractAnalysisData data access interface.

Access to stored data is now done through wrapper object instead of retrieving raw pointers. Also refactored the implementation such that the pure virtual methods that implement the actual data access are now protected, and the public members are now simple non-virtual wrappers that perform common checking and indexing.

Also removed support for negative data frame indexing. It is no longer necessary because the data frame index is now passed to AnalysisDataModuleInterface::pointsAdded(), so it's easy to compute the absolute index of past frames, and additionally it didn't work properly in parallelization schemes. Also removed possibility to access the current frame from pointsAdded(), as it would just create complications to allow access to an incomplete frame.

Part of issue #827.

Change-Id: If932c3f6b8394f8ceacc632fb9d87ec1933f56ca

Use separate class for data storage.

Moved storage implementation from AbstractAnalysisDataStored and parallel storage implementation from AnalysisData to a common AnalysisDataStorage helper class that is used through containment and delegation instead of inheritance.
Makes the code more reusable in eventual implementation of parallel-enabled analysis modules, moves implementation details away from the public interface (AnalysisDataStorage now only needs to be forward-declared in public headers), and clarifies responsibilities in the code.

Support for multipoint data is still not very good (but not worse than it was), but should now be easier to improve. In the future, it could be considered whether to split the unit tests such that storage tests test AnalysisDataStorage directly instead of through the AnalysisData interface, but currently, these classes are quite tightly bound together, so there is not much to gain from separate testing.

Related to issue #827.

Change-Id: l8c34a618893c770bf6d8216500d1061455bee89
All issues except the second (nbsearch) have now fixes pushed for review in gerrit. I'll create a separate issue for the remaining task and close this one once the changes have been reviewed.

Has been merged.

- Project changed from Next-generation analysis tools to Gromacs
- Category set to analysis tools