GROMACS - Feature #2248

Label all SIMD functions as pure/nodiscard

09/14/2017 11:26 PM - Roland Schulz

Status:	New	
Priority:	Normal	
Assignee:		
Category:	mdrun	
Target version:		
Difficulty:	uncategorized	

Description

#2247 would have triggered a compiler warning in Clang if the SIMD function were labeled as

```
[[nodiscard]]
```

(since clang 4) or

```
__attribute__ ((pure))
```

(since at least 3.4).

Questions:
1) Is it worth labeling all SIMD functions/operators to avoid bugs like this in the future?
2) Is it better to label them as pure or nodiscard? Pro pure: could inform the compiler about more than the return value (Does that enable any new optimizations?). Pro nodiscard: non-gcc extension and could be used also for non-pure
3) Should we introduce a gmx_nodiscard/pure? Alternative is to directly use the nodiscard attribute but then we need to "-Wno-attributes" for older compilers which don't have that attribute yet
4) Should the attribute be added to all implementations? Alternative would be to have a separate declaration for all function (which also would contain the doxygen) which has the attribute and have the implementations only have the definition.

Related issues:

Related to GROMACS - Task #2857: Clarify recommended function specifies (cons...	**New**

History

#1 - 09/14/2017 11:29 PM - Roland Schulz

- Description updated

#2 - 11/12/2017 07:31 PM - Szilárd Páll

- Category set to mdrun

I am pro using attributes and I have heard __attribute__ ((pure)) and __attribute__ ((const)) being recommended as the compiler can do more aggressive optimization. Are there no equivalents in clang?

[[nodiscard]] is C++17, though, isn't it?

#3 - 02/04/2019 04:48 AM - Roland Schulz

- Related to Task #2857: Clarify recommended function specifies (constexpr, noexcept, pure) added

#4 - 02/04/2019 04:49 AM - Roland Schulz

clang does support the same attributes.

nodiscard is indeed only C++17 but it would be trivial to define gmx_nodicard which evaluates to nothing for compiler which don't support it. Most our compilers already support it in C++14. Meaning we would get the benefit because Jenkins would be guaranteed to warn.