GROMACS - Feature #2001

add MPI info to the mdrun log header

07/06/2016 04:34 PM - Szilárd Páll

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

Description

It is often beneficial to know what flavor and version of MPI has i) GROMACS been built against ii) is used at runtime, e.g. to quickly identify issues like #1897.

The two can be different both because mdrun can be link against libmpi that will later be upgraded either by OS updater or by a (sloppy) user installing a new MPI over the old one. The launcher can also belong to an MPI install of a different version than the one mdrun is build with.

While all this info *may* be useful, the first things would be to include simple build-time MPI information, I'd say and extend it later.

History

#1 - 07/06/2016 04:39 PM - Szilárd Páll

Would it help is including the path to the MPI lib or MPI compiler as detected?

Another option is extracting the env vars set by mpirun, e.g.:
$ (module load openmpi/1.10.2 && mpirun -np 1 /tmp/test.sh | grep -i "mpi.*version")
OPENMPI_VERSION=1.10.2

$ (module load mpich && mpirun -np 1 /tmp/test.sh | grep -i "mpi.*version")
MPICH_VERSION=3.1.4

However, this requires being able to execute stuff, so this could only be done at runtime (although I imagine for runtime tests mpi.h may define macros that are better suited for version detection).

#2 - 07/06/2016 04:43 PM - Szilárd Páll

Szilárd Páll wrote:
(although I imagine for runtime tests mpi.h may define macros that are better suited for version detection).

Indeed, it does:
OpenMPI

```
/* Major, minor, and release version of Open MPI */
#define OMPI_MAJOR_VERSION 1
#define OMPI_MINOR_VERSION 10
#define OMPI_RELEASE_VERSION 2
```

MPICH

```
/* MPICH_VERSION is the version string. MPICH_NUMVERSION is the
 * numeric version that can be used in numeric comparisons.
 *
 * MPICH_VERSION uses the following format:
 * Version: [MAJ].[MIN].[REV][EXT][EXT_NUMBER]
 * Example: 1.0.7rc1 has
 *          MAJ = 1
 *          MIN = 0
 *          REV = 7
 *          EXT = rc
 *          EXT_NUMBER = 1
 *
 * MPICH_NUMVERSION will convert EXT to a format number:
 *          ALPHA (a) = 0
```

```
 *          BETA (b)  = 1
 *          RC (rc)   = 2
 *          PATCH (p) = 3
 * Regular releases are treated as patch 0
 *
 * Numeric version will have 1 digit for MAJ, 2 digits for MIN, 2
 * digits for REV, 1 digit for EXT and 2 digits for EXT_NUMBER. So,
 * 1.0.7rc1 will have the numeric version 10007201.
 */
#define MPICH_VERSION "3.1.4"
#define MPICH_NUMVERSION 30104300

#define MPICH_RELEASE_TYPE_ALPHA  0
#define MPICH_RELEASE_TYPE_BETA   1
#define MPICH_RELEASE_TYPE_RC     2
#define MPICH_RELEASE_TYPE_PATCH  3
```

#3 - 07/06/2016 05:21 PM - Mark Abraham

Defines are only useful at compile time - one has to call a library function (or query an environment variable set by the library) to get anything at run time.

We do some stuff already in cmake/gmxManageMPI.cmake

One would also want to consider which of OpenMPI, MPICH, MVAPHIC, Intel, Microsoft, Cray and/or IBM one would wish to support - though some are probably MVAPICH derivatives.