MPI Fortran ABI Agenda - August 2023

The topics/decisions I would like to consider are:

1. Can we exclude MPI_Fint and MPI_<handle/status>_{f2c,c2f} from the _first_ ABI ticket? The goal would be to address every aspect of the C ABI except these. We would then address these in a subsequent ticket.

2a. Do we fix the value of MPI_Fint and leave ABI support for Fortran INTEGER sizes not equivalent to C int unspecified, or do we support - by some means - a set of values of MPI_Fint?

2b. If we support multiple values of MPI_Fint, how do we do that? For example, we can use a preprocessor statement such as MPI_ABI_FINT_SIZE that a user can #define to 2, 4, 8, 16 etc. to match the MPI Fortran code.

3. The behavior of MPI_Type_size for MPI_INTEGER, MPI_REAL and MPI_DOUBLE_PRECISION in any language, but especially C, is not well-defined without a prescription of the associated Fortran type behavior. It is not clear how to implement this, because, unlike MPI_Fint, this is implemented in the library, not the header, so it is unclear how we can align this with MPI_ABI_FINT_SIZE.

I sketch one possibility at the bottom, but it has the obvious disadvantage of only working when mpi.h is included. It also violates the fundamental concept of a fixed value for this constant, does not allow MPI_INTEGER to be used via dlsym, etc. In short, it's completely unacceptable from the perspective of what the ABI is supposed to be.

One option I can think of that meets our goals is a new function, e.g., MPI_Abi_set_fortran_sizes(&integer_size, &real_size, &dp_size), that the user is required to call before making the otherwise ambiguous query.

Another option, which is approximately what exists in the ecosystem today is to have a separate implementation of the MPI library for every supported set of Fortran type sizes. Note that because the type sizes can be queried in C, this implies one MPI C library for every Fortran configuration.

Alternatively, we could instruct implementers to implement MPI_Type_size for the relevant types as a call into a function that must be part of the MPI Fortran library, therefore allowing MPI implementations to have a single MPI C library and only build multiple versions of the MPI Fortran library, which includes all of the Fortran symbols but also the Fortran-dependent features of the C API.

4. Is it even reasonable to have a separate MPI Fortran library, or do we insist that the entire MPI implementation be part of a single shared library, in which case MPI implementations are going to ship one MPI library for every supported Fortran configuration?

Thanks,

Jeff

```
#ifdef MPI_ABI_FINT_SIZE
#if MPI_ABI_FINT_SIZE == 2
MPI_INTEGER = MPI_INTEGER2
#elif MPI_ABI_FINT_SIZE == 4
MPI_INTEGER = MPI_INTEGER4
#elif MPI_ABI_FINT_SIZE == 8
MPI_INTEGER = MPI_INTEGER8
#elif MPI_ABI_FINT_SIZE == 16
MPI_INTEGER = MPI_INTEGER16
#else
#error invalid MPI_ABI_FINT_SIZE
#endif
#endif

#ifdef MPI_ABI_FREAL_SIZE
#if MPI_ABI_FREAL_SIZE == 4
MPI_REAL = MPI_REAL4
#elif MPI_ABI_FREAL_SIZE == 8
MPI_REAL = MPI_REAL8
#elif MPI_ABI_FREAL_SIZE == 16
MPI_REAL = MPI_REAL16
#else
#error invalid MPI_ABI_FREAL_SIZE
#endif
#endif
```

(same thing for DOUBLE_PRECISION)