MPI 2.2

William Gropp

Scope of MPI 2.2

- Small changes to the standard. A small change is defined as one that does not break existing *correct* MPI 2.0 or 2.1 user code - either by interface changes or semantic changes - and does not require large implementation changes.
  - What this often means is *adding* something to MPI, such as a new routine or a new basic datatype. Resolving ambiguities can be considered (but don't break programs)
- Note that some topics may be moved to the MPI 2.1 (obvious fix) or MPI 3 discussions (more than minor change)

Process

- Discussion
  - Formation of ad hoc working group to create the …
- Written Proposal
  - Must identify specific pages/lines
  - Must be comprehensive (all relevant lines, including bindings)
  - Should identify possible impact on *incorrect* MPI programs
  - Readings and votes according to rules
- Adopted into MPI 2.1 document to create MPI 2.2 document

3

Topics for MPI 2.2 From Current Errata and Mail Discussions

- Language bindings
  - Use of const
  - Use of restrict
  - New C datatypes
  - Java? Fortran 2003?
  - Info constructor in C++
  - Missing C++ null functions (e.g., CONVERSION_FN_NULL)
- Extensions of Routines
  - MPI_Alltoallx (generalize MPI_Alltoallv)
  - MPI_Alloc_mem behavior on failure
  - Constant blocksized version of reduce_scatter
  - Messages larger than 2GB (e.g., int->MPI_Aint for count?)
  - MPI_IN_PLACE for Reduce_scatter
- Extensions of "constants"
  - MPI_ERR_INFO?
  - C++ SEEK_SET etc and stdio
  - Last error code and last error class
- Open Questions
  - MPI_GROUP_EMPTY
  - MPI_LONG_LONG_INT and MINLOC/MAXLOC
  - Does FILE_GET_VIEW return copies of datatypes?
  - MPI Datatype for sending MPI_Aint and/or MPI_Offset
  - Datatypes and MPI_Probe/Iprobe
  - C++ and interlanguage compatibility
  - Send Buffer Access

4

Additional Topics

- Topics Suggested on Monday
  - Globalization (UTF-8)
  - Wchar_t strings
  - Secure API
  - Require C++ Namespace
  - Collective "v" ops with bound on count
  - Allow more MPI functions to be macros
  - Interaction with multicore/threads
  - Memory allocation
  - Profiling interface (part. Fortran vs C)
  - Mpiexec
  - Synchronized and aggregated print
  - MPI_Offset type extents for datatypes

5

Making Progress

- We need to resolve the open questions (even if that means forwarding them to MPI 3 discussions)
  - Need volunteers to take on open items
- We need to identify self-organizing working groups for larger projects
- Jeff Squyres has volunteered to write up his errata items
- I'll write up mine
- A history of proposals (including failed ones) will be maintained (but not part of the standard)
- We have substantial proposals (see errata page) for
  - Alltoallx
  - MPI_PROC_NULL and epochs
  - Rationale for no MPI_IN_PLACE in MPI_Exscan
  - MPI_IN_PLACE for MPI_Reduce_scatter
  - Send buffer semantics
  - Adding const

6

C++ complex type and interlanguage compatibility

- E.g., Send a complex in Fortran (or C) and receive it in C++
- Proposed text
  - MPI-2, page 276, after line 4, add
  - Advice to users.
  - Most but not all datatypes in each language have corresponding datatypes in other languages. For example, there is no C or Fortran counterpart to the MPI::BOOL or the the MPI::COMPLEX, MPI::DOUBLE_COMPLEX, or MPI:LONG_DOUBLE_COMPLEX. End of advice to users.
- Extending the C++ datatypes to C and Fortran needs to include MPI::BOOL as well as the complex types, and should define what the equivalent types are in C and Fortran. The real issue here is the MPI:F_COMPLEX and completing the list of such routines.

MPI_Alltoallx

- MPI-2, page 164, line 16-30 should read:
  7.3.5. Generalized All-to-all Functions
  One of the basic data movement operations needed in parallel signal processing is the 2-D matrix transpose. This operation has motivated two generalizations of the MPI_ALLTOALLV function. These new collective operations are MPI_ALLTOALLW and MPI_ALLTOALLX; the ``W'' indicates that it is an extension to MPI_ALLTOALLV, and ``X'' indicates that it is an extension to MPI_ALLTOALLW. MPI_ALLTOALLX is the most general form of All-to-all. Like MPI_TYPE_CREATE_STRUCT, the most general type constructor, MPI_ALLTOALLW and MPI_ALLTOALLX allow separate specification of count, displacement and datatype. In addition, to allow maximum flexibility, the displacement of blocks within the send and receive buffers is specified in bytes. In MPI_ALLTOALLW, these displacements are specified as integer arguments and in MPI_ALLTOALLX they are specified as address integer.

- Rationale. The MPI_ALLTOALLW function generalizes several MPI functions by carefully selecting the input arguments. For example, by making all but one process have sendcounts[i] = 0, this achieves an MPI_SCATTERW function. MPI_ALLTOALLX allows the usage of MPI_BOTTOM as buffer argument and defining the different buffer location via the displacement arguments rather than only via different