A Preview of MPI 3.0: The Shape of Things to Come

Manjunath Gorentla Venkata
manjugv@ornl.gov

Joshua Hursey
hurseyjj@ornl.gov (or jhursey@uwlax.edu)
Overview of Seminar Series

• Monday, June 25 - 3-4 pm:
  – MPI Process (brief)
  – Timeline to 3.0
  – MPI 3.0 Fortran Bindings
  – MPI 2.2

• Tuesday, June 26 - 3-4 pm
  – Collectives:
    • Neighborhood
    • Nonblocking
  – Communicator Creation:
    • Noncollective
    • Nonblocking duplication

• Thursday, June 28 - 3-4 pm
  – MPI Matched Probe/Recv
  – RMA / One-sided enhancements
  – Tool Interfaces
  – MPI <next>
    • Fault Tolerance
    • Hybrid, collectives, …
Feedback and Discussion

• We want lots of feedback from you!
  • What features are useful?
  • What features would you like to know more about?
  • What features are not useful?
  • What features are missing?

• Please interject with questions as we go

• Please send us comments and questions afterward
  • We can also help connect you with prototypes and researchers

• This will help us better support you on OLCF machines
  • Determine areas to focus research and development efforts
  • Prototype ➔ Production-quality, scalable algorithms
Overview of Seminar Series

• Monday, June 25 - 3-4 pm:
  – MPI Process (brief)
  – Timeline to 3.0
  – MPI 3.0 Fortran Bindings
  – MPI 2.2
The MPI Process

- MPI Standard: Open standard ratified by the MPI Forum
- MPI Forum Standardization Body
  - Started meeting again in 2007 after a 10 year hiatus
  - Meets 4-5 times a year (prior to this year, 6-7 times a year)
- Process
  - Each organization gets 1 vote (Must attend 2 out of last 3 meetings to vote)
  - Proposals must go through a long process before standardization
    - At least 3 meetings: First Reading ➔ First Vote ➔ Second Vote
    - Simple majority vote required to pass
- Anyone can attend: (it's a lot of fun … really … well somewhat)
  - Manju is the representative from ORNL
MPI Standard Timeline

• MPI Versions
  • 1.0 – 1994
  • 1.1 – 1995
  • 1.2 – 1997
  • 2.0 – 1997
  • 1.3 – 2008
  • 2.1 – 2008
  • 2.2 – 2009: Current - Combined 1.X and 2.X documents
  • 3.0 – 2012: In preparation
MPI Standard Timeline

• MPI Standard 3.0
  • July 2012 – Chicago, IL
    • Last of the Second Votes for 3.0 proposals
    • Final chapter edits (integrating proposals)
    • Prepare a 'Draft Standard' for circulation
  • September 2012 – Vienna, Austria
    • Formal Reading of the whole standard*
      *We might do this over the phone and release 3.0 in September (to be decided in July)
  • December 2012 – San Jose, CA
    • Final Chapter vote
    • Release 3.0
MPI Standard Implementation Timeline

- Prototype implementations were required for most proposals
  - Some prototypes are not really for public consumption
- Implementation availability is on a per-feature basis
  - We will discuss availability as we mention features
- If the feature is something you want access to let us know
  - We will get you in contact with the appropriate people
  - We will also push to get these features into the various MPI implementations on OLCF machines
- Generally, it may take another year or so before all of these features are widely available
Overview of Seminar Series

• Monday, June 25 - 3-4 pm:
  – MPI Process (brief)
  – Timeline to 3.0
  – **MPI 3.0 Fortran Bindings**
  – MPI 2.2
MPI 3.0 Fortran Bindings

• The way of the future: use mpi_f08

• Requirements Highlights:
  • Comply with Fortran standard (for the first time)
    • Fortran 2008 Compliance
      • MPI Forum worked together with the Fortran Standards Technical Committee
        http://www.j3-fortran.org/
  • Compile-time subroutine parameter type checking
  • "ierr" is now an optional argument!
  • Convenient upgrade migration path for users
  • Send/Recv sub-arrays
  • Correct asynchronous support

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings:
Subroutine Parameter Type Safety

• All parameter types are checked
  • Pass the wrong type or skip a required parameter = Compiler error

• MPI handles are uniquely typed
  • MPI handles are derived types: TYPE(MPI_Comm)
  • Pass MPI_Datatype to an MPI_Comm = Compiler error

• Examples:
  • call MPI_Send(buf, count, datatype, dest, comm, tag, ierr)
  • call MPI_Send(buf, count, datatype, dest, tag, comm, ierr)
  • call MPI_Send(buf, count, datatype, dest, comm, ierr)

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings: ierr is now optional!

• ierr argument to MPI subroutines is now optional!
• It is the only optional argument (at the moment?)

• Examples:
  • call MPI_Send(buf, count, datatype, dest, tag, comm, ierr)
  • call MPI_Send(buf, count, datatype, dest, tag, comm)

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings: Interpretability & Backwards Compatibility

• The way of the future: `use mpi_f08`
  • "include mpif.h" and "use mpi" will not go away
  • No backwards-incompatible changes added to the standard

• Interpretability of all three in a single application
  • 1 per subroutine
  • Easy conversion between new and old Fortran handles

• Applications (libraries) can gradually adopt "use mpi_f08"
  • Requirement of a convenient migration path for users

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings: Interpretability & Backwards Compatibility

```fortran
subroutine legacy_subroutine(oldcomm, newcomm)
   include 'mpif.h'
   integer oldcomm, newcomm

   call new_subroutine(oldcomm, newcomm)
   call MPI_Comm_send( ..., newcomm)
end subroutine

subroutine new_subroutine(oldcomm, newcomm)
   use mpi_f08
   integer oldcomm, newcomm
   type(MPI_Comm) oldcomm_f08, newcomm_f08

   oldcomm_f08%MPI_VAL = oldcomm
   call MPI_Comm_dup(oldcomm_f08, newcomm_f08)
   newcomm = newcomm_f08%MPI_VAL
end subroutine
```

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings: Send/Recv Sub-Arrays

• Send and receive sub-arrays
  
  \[
  \text{call MPI\_Irecv(Array(1,:), \ldots, request, \ldots)}
  \]

• Currently you would need to build a new datatype for this
  
  \[
  \text{call MPI\_Type\_create\_subarray(\ldots, dt, ierr)}
  \]
  \[
  \text{call MPI\_Irecv(Array(:,,:), 1, dt, \ldots, request, \ldots)}
  \]

• Requires compiler support
  
  • Estimated 1-2 years

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings: Correct MPI Asynchronous Support

- Guarantee of correct asynchronous operations
  
  ```fortran
  call MPI_Irecv(buffer, ..., request, ...)  
  ...
  call MPI_Wait(request, status)
  a = buffer(1)
  ```

- Problem stems from: Fortran has no pointer aliasing
  - Compilers tend to aggressively re-order code
  - Compiler can move the code `a=buffer(1)` above the `MPI_Wait()`

- Fixed with some new Fortran language constructs
  - `DIMENSION(..)` and `ASYNCHRONOUS` attribute for choice buffers

- Requires compiler support
  - Estimated 1-3 years

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings: Availability

- Open MPI Prototype
  - Available today in the trunk (scheduled part of the 1.7 release series)
  - `mpifort` wrapper compiler replaces `mpif77` and `mpif90`
    - `mpif77` and `mpif90` still exist for backwards compatibility... for a while
  - `ompi_info` will indicate the f08 features (not) supported by the compiler

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings: Availability

• Open MPI Prototype
  • **Currently supports:**
    • Enhanced Type Safety
    • Optional ierr parameter
    • Interoperability of mpif.h, use mpi, and use mpi_f08 together in a single application
  • **Eventually will fully support:** (write to your favorite compiler vendor!)
    • Send/Recv Fortran array subsection (1-2 years)
    • Correct MPI asynchronous support (1-3 years)
  • Latest testing shows:
    • **gfortran:** Does not support any of the mpi_f08 stuff
    • **Intel:** Supports everything except the "Eventually" clauses
    • **PGI/Absoft:** Supports mpi_f08, but not quite everything (see ompi_info for details)

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
MPI 3.0 Fortran Bindings: Highlights

• The way of the future: use mpi_f08
  • Comply with Fortran standard (for the first time)
    • Fortran 2008 Compliance
      • MPI Forum worked together with the Fortran Standards Technical Committee http://www.j3-fortran.org/
  • Compile-time subroutine parameter type checking
  • "ierr" is now an optional argument!
  • Convenient upgrade migration path for users
  • Send/Recv sub-arrays
  • Correct asynchronous support

Thanks to Craig Rasmussen (LANL), Rolf Rabenseifner (HLRS), Jeff Squyres (Cisco Systems)
Overview of Seminar Series

- **Monday, June 25 - 3-4 pm:**
  - MPI Process (brief)
  - Timeline to 3.0
  - MPI 3.0 Fortran Bindings
  - **MPI 2.2**