Just Writing a Standard is Not Enough!

Martin Schulz

Technical University of Munich
Department of Informatics
Chair of the MPI Forum

MVAPICH User’s Group Meeting
Columbus, Ohio, USA
August 2018
Technical University of Munich

Located in Munich, Germany
- Founded in 1868
- Around 45,000-50,000 students
- All Engineering disciplines
- “University of Excellence”

Multiple locations
- Core is downtown Munich
- Hospital also located downtown
- Life sciences in Weihenstephan
- Largest campus: Garching (NE of the city)
Department of Informatics
Department of Informatics
- We have slides! -
Department of Informatics
SuperMUC-NG – End of 2018

Specs
- 26.7 Pflop/s
- 700 Tbyte main memory and
- 70 Pbyte disk storage
- 6,400 Lenovo ThinkSystem nodes with Intel Xeon processors
- 300,000 compute cores
- Intel Omni-Path interconnects
- Hot water cooled

HPC + Cloud
- Usage of own and individual virtual machines (integrated cloud)
- Pre- and post-processing with user’s individual software
- Integrated development, ability to use familiar software and tools
- Remote visualization and integration to V2C
Just Writing a Standard is Not Enough!

Martin Schulz
Technical University of Munich
Department of Informatics
Chair of the MPI Forum

MVAPICH User’s Group Meeting
Columbus, Ohio, USA
August 2018
Just Writing a Standard is Not Enough!
- or –
What Made MPI What it is Today?

Martin Schulz

Technical University of Munich
Department of Informatics
Chair of the MPI Forum

MVAPICH User’s Group Meeting
Columbus, Ohio, USA
August 2018
MPI is (over) 25 Years Old

Celebrating 25 Years of MPI, Special Event at EuroMPI/USA 2017, Chicago, IL, USA
The Starting Point

1991 - 1992

- The MPI effort began in the summer of 1991 when a small group of researchers started discussions at a mountain retreat in Austria.
  - IBM Oberlech Conference
  - Tony Hey, Rolf Hempel, Ulrich Trottenberg, JD
- Message passing was in the air
  - Intel NX, Express, Zipcode, PARMACS, IBM EUI/CCL, PVM, P4, Occam, Linda, ...
- Many groups pursuing ideas
- Concern there would be European and competing US efforts
- The community felt the need to have a standard message passing interface.
  - We were developing ScaLAPACK and needed a portable way to do MP

Slide from Jack Dongarra, Talk given at “Celebrating 25 Years of MPI”
MPI Kept on Growing

- MPI 1.0 May 1994 – 228 pages
- MPI 1.1 Nov 1995 – 238 pages (128 fns)
- MPI 2.0 Nov 1997 – 608 pages

**10 year break**

- MPI 2.1 June 2008 – 608 pages
- MPI 2.2 Sep 2009 – 647 pages
- MPI 3.0 Sep 2012 – 852 pages (430 fns)
- MPI 3.1 June 2015 – 868 pages

#Words


Slide by Adam Moody
MPI is Alive and Kicking

MPI grew to be the dominant programming standard in HPC
- Used by vast majority of HPC applications
- Runtime portability layer for other programming models
- Part of basically any HPC procurement

Example: Survey among ECP projects in the US (conducted by OMPI-X)
- Basically all application projects use MPI
- 93% expect to use MPI at Exascale
- 57% of middleware projects use MPI

Stable community participation
- Vendors and open source MPI developers
- Tools community
- Good participation in community events, like BoFs

Part of many CS curricula
What Made MPI the Dominating HPC Standard?
What Made MPI the Dominating HPC Standard?

- Targeting an Important Problem at the Right Time
The Starting Point

1991 - 1992

- The MPI effort began in the summer of 1991 when a small group of researchers started discussions at a mountain retreat in Austria.
  - IBM Oberlech Conference
  - Tony Hey, Rolf Hempel, Ulrich Trottenberg, JD
- Message passing was in the air
  - Intel NX, Express, Zipcode, PARMACS, IBM EUI/CCL, PVM, P4, Occam, Linda, ...
- Many groups pursuing ideas
- Concern there would be European and competing US efforts
- The community felt the need to have a standard message passing interface.
  - We were developing ScaLAPACK and needed a portable way to do MP
What Made MPI the Dominating HPC Standard?

- Targeting an Important Problem at the Right Time
- Functionality
Starting Point: Simple, Complimentary Concepts

Minimal set of functions
- Only 6 functions needed to write a program
- Send/Recv as basic primitives

Nonblocking operations to support overlap
- Initially limited to P2P
- Matches P2P functions
- Returns completion object

Collectives to support scalability
- Most common operations
- Fan-in, fan-out and fan-in-out supported
- Reductions with user defined operations

Persistence to support optimization
- Initially limited to P2P
- Definition and launch of repeating operations
Consistent Design Concepts

Built on few, but cross cutting abstractions
• Strong goal to be consistent

Communication contexts in the form of communicators
• Pervasive throughout the standard

Generalizable data types
• Flexible data specification for any routine

Orthogonality
• Some functionality on multiple versions (e.g., blocking, non-blocking, persistent)
• Increases number of functions
• But provides ease of use

Independence of runtime system
• Enables easier portability
Some of the Newer Additions up to MPI 3.1

Nonblocking collectives
- Closing an obvious orthogonality issue
- Same basic wait/test semantics as P2P

Neighborhood collectives
- Scalable collectives
- Only specify direct neighbors

Remote Memory Access (RMA)
- Adds a second, orthogonal communication mechanism
- Introduced in 2.0, improved in 3.0

MPIIO

Tool Support
- Profiling interface from the start
- MPI Tool Information Interface since MPI 3.0
- Role model for other standards
MPI 4.0 is Coming Up Soon!
MPI 4.0 is Coming Up Soon!

New/Ratified Features for MPI 4.0

Assertions for message traffic to guide optimization
  • Can state that an application doesn’t use wildcards
  • Enables traffic optimizations
  • Great opportunities for implementations

Remove info key propagation on communicator duplication
  • New function: MPI_Comm_idup_with_info
  • Better control over properties attached to communicator

Clarification of what it means to query the info object attached to an MPI object

Deprecation of send cancel
  • Long overdue 😊

Small fixes to the MPI Tools Information Interface
Anticipated Features for MPI 4.0

Persistent version of all collectives
• Setup of repeating collectives, similar to P2P
• Kickoff with a single command
• Impact 1: new optimization options
• Impact 2: opens new algorithmic options to reduces ordering constraints

Clarification of abort behavior to enhance ability to be fault tolerant
• No longer require all of MPI fails
• Enables MPI implementations to limit failures to communicators
• General failures are mapped to MPI_COMM_SELF
  – WARNING: BACKWARDS INCOMPATIBLE
• First step towards support for fault tolerance

Non blocking constructor/destructor for RMA operations
Under Discussion for MPI 4.0

MPI_T Events: additional tool interface based on callbacks
  • Extensions of the MPI_T concept
  • Provides tracing capabilities for MPI internal information

Clarification across the entire standard of MPI process vs. OS clarification
  • Make it explicit that implementations with “1 MPI process = 1 thread” are allowed

Large count support for transmission and I/O of large messages/chunks
  • Requested by user community
  • Targets communication and file I/O

Handling of native types in host languages
  • Triggered by need to support FP16
  • General solution

Further asynchronous and persistent operations to strengthen orthogonality
Road Map Towards MPI 4.0
(Tentative)

Draft Standard at SC 18
• All items passed by the forum by September 2018 meeting
• Intermediate Snapshot for Community Feedback

Draft Release Standard at SC 19
• All items passed by the forum by September 2019 meeting
• "Beta-Standard" for MPI 4.0
  – Feature freeze of MPI 4.0 for Community Feedback
  – Start work on “MPI 4.0-next”

Standard before ISC 2020
What Made MPI the Dominating HPC Standard?

- Targeting an Important Problem at the Right Time
- Functionality
  - A Dedicated and Open Team
Participants in the Forum over the Years

A. Gordon Smith
Abhinav Vishnu
Adam Greenberg
Adam Moody
Akshay Venkatesh
Al Geist
Alan Mainwaring
Alan Sussman
Albert Cheng
Alex Ho
Alexander Supalov
Alexey Cheptsov
Alice Koniges
Abbaj Singh
Amin Hassan
Amit Maimida
Andrew Lumsdaine
Andrew Sherman
Anh Vo
Anna Rounbeiler
Anne Elster
Anthony Skjellum
Antonio J. Pena
Arch Robison
Arndam Saha
Arkady Kanevsky
Amo Candel
Arthur Maccabe
Atsushi Hori
Aurelien Bouteiller
Avneesh Pant
Balazs Gerofi
Bill Long
Bill Nitzberg
Bill Saphir
Bin Jia
Bob Knighten
Bob Leary
Bob Madahar
Boris Protopopov
Brian Barrett
Brian McCandless
Brian Smith
Brice Goglin
Bronis R. de Supinski
C.T. Howard Ho
Charles Archer

Charles Mosher
Christian Bell
Christian Siebert
Christof Klauskeuer
Christos Kavoukli
Chulho Kim
Craig Fischberg
Craig Rasmussen
Dan Nessett
Daniel Fye
Daniel Holmes
Darius Buntinas
Damen Sanders
Dave Goodell
Dave Wright
David Diucci
David Goodell
David Greenberg
David Solt
David Taylor
David Walker
Davide Rosseliti
Denis Nagomy
Dennis Cottet
Dennis Weeks
Devendr Bureddy
Dick Teumann
Dieter Kranzmueller
Dmitry Dumov
Don Heller
Doug Doeffter
Douglas Miller
Dries Kimpe
Ed Anderson
Ed Benson
Edgar Gabriel
Edric Ellis
Elsie Pierce
Enqiang Zhou
Erez Haba
Eric Bansczc
Eric Brunner
Eric Lantz
Eric Salo
Eric Shanakan

Ewing Lusk
Fabian Tiller
Fred Shirley
Gabor Dozza
Gallem Shipman
Gary Howell
Geoffroy Vallee
George Boslica
George Carr
Glad Shainer
Graet Asflak
Greg Bornevetsky
Greg Bums
Greg Tensa
Guillaume Mercier
Hans-Christian Hoppe
Haf Subramoni
Haith Nig
Heid Poxon
Hideyuki Jitsumoto
Howard Palmer
Howard Pritchard
Hubert Ritzdorf
Hubertus Franke
Huwei Lu
Huseyn S. Yildiz
Ian Foster
Ildan Gleeninning
Ian Stockdale
Ignacio Laguna
Jack Dongama
James Cowrie
James Dinan
James Kohl
James Pruye
Jarek Nieplucha
Jay Lofstead
Jean-Pierre Prost
Jed Brown
Jeff Brown
Jeff Hammonds
Jeff Squires
Jeffrey Brown
Jeffrey M. Squires
Jennifer Hemett-Skjellum

Jerrell Watts
Jesper Larson Traeff
Jesper Larson Truff
Jim Cownie
Jim Feeney
Jithin Jose
Joe Baron
Joe Rieken
Joe Clark
John Hagedom
John Kapenga
John May
Jon Flower
Jonathan Carter
Hoskeuskey
Huisa Lodd
Judith Devaney
Junchoa Zhang
Kannan Narasihman
Kein Feind
Karl Kesselman
Kathryn Mororo
Kei Harada
Keita Terezhishi
Keith Underwood
Ken Raffennetti
Kento Sato
Koichi Kishoshi
Krishna Kandallla
Kuninobu Sasaki
Lance Shuler
Laurie Costello
Leahie Hart
Lloyd Lewins
Lind Clarke
Maciej Brodowicz
Manjuschana Ganenta
Manojkumar Krishnan
Manuel Ujuldon
Marc Snir
Marc-Andre Hammans
Margaret Cahir

Mario Laura
Mark Falcon
Mark Truff
Mark Pagel
Mark Sears
Martin Schulz
Marty Itzkowitz
Marydel Tholbun
Masamichi Takagi
Matthew Koop
Michael Blocksome
Michael Knobloch
Min Xie
Miron Livny
Mohamed Chaanawi
Naoki Sueyasu
Nathan DeBardeleben
Nathan Davis
Nathan Heim
Nicholas Radcliffe
Nick M. Maclaren
Nick Nevin
Nobutoshi Sagawa
Nysal Jan
Oliver McByran
Paddy Gileos
Pang Chen
Parkson Wong
Patrick Geoffray
Paul Pierece
Pavan Balaji
Pavel Shamas
Perry Partow
Peter Bradby
Peter Brennan
Peter Madams
Peter Ossadnik
Peter Pacheco
Peter Rigsbee
Phil McKinley
Pratap Pattnaik
Prusotham V. Banglore
Quincey Koziol
Raffele Giuseppe Solca
Raghunath Raja
Raghunath Raja

Rahul Sarekark
Rainer Keller
Raja Daoind
Rajevec Thakur
Rajesh Bordawekar
Rancher Keller
Reinhold Bader
Richard Barrett
Richard Frost
Richard L. Graham
Richard Littlefield
Richard Teumann
Rob Bjomson
Rob Ross
Robert Babb
Robert Blackmore
Robert G. Voigt
Robert George
Robert Hartson
Robert Tomlinson
Rolf Hempel
Rolf Rabenseifner
Rolf Riesen
Rolf Vandevea
Rolf vandeVaart
Ron Brightwell
Ron Oldfield
Ryan E. Grant
Sadan Alaf
Sam Fineberg
Sameer Kumar
Sangmin Seo
Sanjay Ranka
Sayantan Sur
Scott Berryma
Scott McMillan
Shane Herteb
Shinji Sumimoto
Simon Tsang
Sreev Panduri
Stephen Lederman
Suresh Damodaran-Kamal
Susan Kraus
Suhnu Krauss
Swann Peramau
Takahiro Kawashima
Takeshi Nantat
Tatsuya Abe
Terry Donta
Terry Jones
Thom Mahon
Thomas Francos
Thomas Herault
Tim Muray
Timo Schiender
Timothy I. Mattox
Todd Gamblin
Tom Haupt
Tom Henderson
Tom Robey
Tomotake Nakamura
Tomoya Adachi
Tony Skjellum
Torsten Hoefker
Tye McClarty

Venkat Vishwanath
Vince Fernando
Vino Tippiraju
Weikuan Yu
Wesley Bland
William Gropp
Xin Zhao
Yann Kalemkarlian
Ying Chen
Yohann Burette
Yong Cho
Yohann Burette
Yong Cho
Yoohoo Park
Yuichi Tssuijita
Yutaka Ishikawa
Zhenqian Cui
Zhiyuan Lu
Ziyang Lu

Lederman
Kamal
The MPI Forum  

https://www.mpi-forum.org/

Standardization body for MPI
• Discusses additions and new directions
• Oversees the correctness and quality of the standard
• Represents MPI to the community

Open membership
• Any organization is welcome to participate
• Consists of working groups and the actual MPI forum
• Physical meetings 4 times each year (3 in the US, one with EuroMPI conference)
  – Working groups meet between forum meetings (via phone)
  – Plenary/full forum work is done mostly at the physical meetings
• Voting rights depend on attendance
  – An organization has to be present two out of the last three meetings (incl. the current one) to be eligible to vote

Technical work driven by the working groups
• https://www.mpi-forum.org/mpi-40/
The Bulk of Work is in the Working Groups

Collective Communication, Topology, Communicators, Groups
  • Torsten Hoefler, Andrew Lumsdaine and Anthony Skjellum

Fault Tolerance
  • Wesley Bland, Aurélien Bouteiller and Rich Graham

HW Topologies
  • Guillaume Mercier

Hybrid Programming
  • Pavan Balaji and Jim Dinan

Big Count
  • Jeff Hammond and Anthony Skjellum

Persistence
  • Anthony Skjellum

Point to Point Communication
  • Rich Graham and Dan Holmes

Remote Memory Access
  • Bill Gropp and Rajeev Thakur

Tools
  • Kathryn Mohror and Marc-Andre Hermanns
What Made MPI the Dominating HPC Standard?

✓ Targeting an Important Problem at the Right Time

✓ Functionality

✓ A Dedicated and Open Team

• Consensus Driven Process
Typical Way New Features Get Added to MPI

1. New items brought to a matching working group for discussion with a focus on established techniques ripe for standardization

2. Creation of preliminary proposal

3. Socializing of idea driven by the WG
   Through community discussions, user feedback, publications, …
   Development of full proposal
   In many cases accompanied with prototype development work

4. MPI forum reading/voting process
   One reading
   Two votes
   Slow and consensus driven process

5. Once enough topics are completed:
   Publication of a new standard
What Made MPI the Dominating HPC Standard?

✓ Targeting an Important Problem at the Right Time
✓ Functionality
✓ A Dedicated and Open Team
✓ Consensus Driven Process

• Multiple Active Open Source Reference Implementations
Multiple Reference Implementations are Crucial

In MPI we have several reference implementation

• MPICH and Open MPI as cores
• MVAPICH derived from MPICH, but with its own character
• Alternative implementations, like MPC

All in public repositories

• Enables active development
• Encourages contributions

Advantages

• Competition is good for performance
• Alternative deployments as risk mitigation
• Ability to modify for research
<table>
<thead>
<tr>
<th>Feature</th>
<th>MPICH</th>
<th>MVAPICH</th>
<th>Open MPI</th>
<th>Cray MPI</th>
<th>Tianhe MPI</th>
<th>Intel MPI</th>
<th>IBM BG/Q MPI(^1)</th>
<th>IBM PE MPICH(^2)</th>
<th>IBM Platform</th>
<th>SGI MPI</th>
<th>Fujitsu MPI</th>
<th>MS MPI</th>
<th>MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBC</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Q4’15</td>
</tr>
<tr>
<td>Nbrhood collectives</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Q4’15</td>
</tr>
<tr>
<td>RMA</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Shared memory</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Tools Interface</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>*</td>
<td>Q4’16</td>
</tr>
<tr>
<td>Comm-creat group</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>F08 Bindings</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q2’16</td>
<td></td>
</tr>
<tr>
<td>New Datatypes</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q4’15</td>
<td></td>
</tr>
<tr>
<td>Large Counts</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q2’16</td>
<td></td>
</tr>
<tr>
<td>Matched Probe</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q2’16</td>
<td></td>
</tr>
<tr>
<td>NBC I/O</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q2’16</td>
<td></td>
</tr>
</tbody>
</table>

1 Open Source but unsupported  
2 No MPI_T variables exposed  
* Under development  
(*) Partly done

Slide by Pavan Balaji
## MPI-3.1 Impl. as of November 2015

<table>
<thead>
<tr>
<th></th>
<th>MPICH</th>
<th>MVAPICH</th>
<th>Open MPI</th>
<th>Cray MPI</th>
<th>Tianhe MPI</th>
<th>Intel MPI</th>
<th>IBM BG/Q MPI(^1)</th>
<th>IBM PE MPICH(^2)</th>
<th>IBM Platform</th>
<th>SGI MPI</th>
<th>Fujitsu u MPI</th>
<th>MS MPI</th>
<th>MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>(*) Q4’15</td>
<td></td>
</tr>
<tr>
<td>Nbrhood collectives</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Q4’15</td>
<td></td>
</tr>
<tr>
<td>RMA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Shared memory</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td><strong>Tools Interface</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>*</td>
<td>Q4’16</td>
</tr>
<tr>
<td>Comm-creat group</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>F08 Bindings</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Q2’16</td>
<td></td>
</tr>
<tr>
<td>New Datatypes</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Q4’15</td>
<td></td>
</tr>
<tr>
<td>Large Counts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Q2’16</td>
<td></td>
</tr>
<tr>
<td>Matched Probe</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Q2’16</td>
<td></td>
</tr>
<tr>
<td>NBC I/O</td>
<td>✓</td>
<td>Q1’16</td>
<td>✓</td>
<td>Q4’15</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Q2’16</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) Open Source but unsupported  \(^2\) No MPI_T variables exposed  * Under development  (*) Partly done

Slide by Pavan Balaji
The MPI Tool Information Interface

Short name: the MPI_T interface
• Provide introspection into the MPI layer
• Export internal performance information
• Enable standardized access for tools
• Allow for implementation specific information

Provide access to MPI internal information
• Configuration and control information
• Performance information
• Debugging information

Standardized access to this information
• Build on top of the success of the PMPI interface
• Integrated MPI_T into the MPI standard as of MPI 3.0
The MPI_T Challenge

Which variables exist can differ between …
• … MPI implementations
• … versions of the MPI library (debug vs. productions)
• … executions of the same application/MPI library
• Libraries can decide not to provide any variables

Example: Performance Variables:
• Users need to query existing variables
• MPI provides names, descriptions, and metadata
### Performance Variables

**Found 25 performance variables**

**Found 25 performance variables with verbosity \( \leq \text{D/A-9} \)**

<table>
<thead>
<tr>
<th>Variable</th>
<th>VRB</th>
<th>Class</th>
<th>Type</th>
<th>Bind</th>
<th>R/O</th>
<th>CNT</th>
<th>ATM</th>
</tr>
</thead>
<tbody>
<tr>
<td>posted_recvq_length</td>
<td>U/D-2</td>
<td>LEVEL</td>
<td>UINT</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>unexpected_recvq_length</td>
<td>U/D-2</td>
<td>LEVEL</td>
<td>UINT</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>posted_recvq_match_attempts</td>
<td>U/D-2</td>
<td>COUNTER</td>
<td>UNKNOW</td>
<td>n/a</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>unexpected_recvq_match_attempts</td>
<td>U/D-2</td>
<td>COUNTER</td>
<td>UNKNOW</td>
<td>n/a</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>time_failed_matching_postedq</td>
<td>U/D-2</td>
<td>TIMER</td>
<td>DOUBLE</td>
<td>n/a</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>time_matching_unexpectedq</td>
<td>U/D-2</td>
<td>TIMER</td>
<td>DOUBLE</td>
<td>n/a</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>unexpected_recvq_buffer_size</td>
<td>U/D-2</td>
<td>LEVEL</td>
<td>UNKNOW</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mem_allocated</td>
<td>U/B-1</td>
<td>LEVEL</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mem_allocated</td>
<td>U/B-1</td>
<td>HIGHWAT</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_progress_poll_count</td>
<td>D/B-7</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>all_bcast_binomial</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>coll_bcast_scatterDoubling_allgather</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>coll_bcast_scatter_ring_allgather</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_num_2level_comm_requests</td>
<td>U/D-2</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_num_2level_comm_success</td>
<td>U/D-2</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_num_shmem_coll_calls</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_binomial</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_scatterDoubling_allgather</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_scatter_ring_allgather</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_scatter_ring_allgather_shm</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_shmem</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_knomialoint</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_knomialoint</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_mcast_internode</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_pipeline</td>
<td>T/B-4</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
... And Continued to Expand Them

### MPI_T Variable List
- MPI Thread support: MPI_THREAD_SINGLE
- MPI_T Thread support: MPI_THREAD_MULTIPLE

---

#### Performance Variables
---

<table>
<thead>
<tr>
<th>Variable</th>
<th>VRB</th>
<th>Class</th>
<th>Type</th>
<th>Bind</th>
<th>R/O</th>
<th>CNT</th>
<th>ATM</th>
</tr>
</thead>
<tbody>
<tr>
<td>mem_allocated</td>
<td>U/B-1</td>
<td>LEVEL</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>num_malloc_calls</td>
<td>U/B-1</td>
<td>HIGHWAY</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>num calloc_calls</td>
<td>T/D-5</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>num_memalign_calls</td>
<td>T/D-5</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>num_strdup_calls</td>
<td>T/D-5</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>num_realloc_calls</td>
<td>T/D-5</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>num_free_calls</td>
<td>T/D-5</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>num_memalign_free_calls</td>
<td>T/D-5</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_num_2level_comm_requests</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_num_2level_comm_success</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_num_shmem_coll_calls</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mpir_progress_poll</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_smp_read_progress_poll</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_smp_write_progress_poll</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_smp_read_progress_poll_success</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_smp_write_progress_poll_success</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>rdma_ud_retransmissions</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_binomial</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_scatter_doubling_allgather</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_scatter_allgather</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_scatter_allgather_shm</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_shmem</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_knomial_internode</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcast_knomial_intranode</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mv2_coll_bcasticast_interode</td>
<td>U/B-1</td>
<td>COUNTER</td>
<td>ULLONG</td>
<td>n/a</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>
... And Continued to Expand Them
# Examples of Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>posted_recvq_match</strong></td>
<td>Counts how many times the queue for receiving expected messages is read.</td>
</tr>
<tr>
<td><strong>unexpected_recvq_match</strong></td>
<td>Counts how many times the queue for receiving unexpected messages is read.</td>
</tr>
<tr>
<td><strong>progress_poll_count</strong></td>
<td>Counts how many times the application polls the progress of a communication. The higher the value, the more CPU time is spent in polling.</td>
</tr>
<tr>
<td><strong>mem_allocated_level</strong></td>
<td>Gives the instantaneous memory usage by the library in bytes.</td>
</tr>
<tr>
<td><strong>mem_allocated_highwater</strong></td>
<td>Gives the maximum number of bytes ever allocated by the MPI library at a given process for the duration of the application.</td>
</tr>
<tr>
<td><strong>coll_bcast_binom</strong></td>
<td>Counts how many of the MPI broadcast collective calls use the Binomial algorithm during an application run.</td>
</tr>
<tr>
<td><strong>num_shmem_coll</strong></td>
<td>Counts how many of the collective communication calls are using shared memory.</td>
</tr>
<tr>
<td><strong>coll_bcast_shmem</strong></td>
<td>Counts how many of the MPI broadcast communication calls are shared memory based collectives.</td>
</tr>
</tbody>
</table>
Example: Receive Queues for NAS BT
Example: Memory Consumptions for NEK5000
Example: Validation of Message Queue Stress Test

Exp Queue Stress Test - Queue Match Attempts
(x:= #messages per sending rank )

Max posted_recvq_match_attempts
Max unexpected_recvq_match_attempts

By K. Kraljic, TUM, Measured with Gyan: https://github.com/llnl/mpi-tools
MPI_T events: Callback-driven event information

Motivation
- PMPI does not provide access to MPI internal state information
- MPI_T performance variables only show aggregated information

New interface to query available runtime event types
- Follows the MPI_T variable approach
- No specific event types mandated
- Event structure can be inferred at runtime

Register callback functions to be called by the MPI runtime
- Runtime may defer callback invocation (tool can query event time)
- Runtime may reduce restrictions on callback functions per invocation
- Callback can query event information individually or copy data en bulk

Targeted for MPI 4.0
What Made MPI the Dominating HPC Standard?

✓ Targeting an Important Problem at the Right Time
✓ Functionality
✓ A Dedicated and Open Team
✓ Consensus Driven Process
✓ Multiple Active Open Source Reference Implementations
  • Adjust to Changing Landscapes
New Worlds Pose New Requirements

Support for new application areas
• BigData, Deep Learning, AI
  • Starting point for FP 16 discussion
• Runtime for loosely coupled systems
  • Requires some form of fault tolerance
• New workflows, like in-situ, ensembles, scale-bridging
  • MPI Sessions as one approach under discussion
• Runtime for novel runtime systems (e.g., tasking systems)

MPI is very static in a growing dynamic world
• Require some form of support for malleability
• MPI Sessions could be the right gateway
• Need to teach users that MPI does not equal SPMD

Modern, composable tooling stack
• Integration of multiple „tools“ or system software elements
  • FT management, power management, system monitoring
• “QMPI” efforts for a new profiling interface underway

Must stick with standardizing state of the art
Final Thoughts

Standards are more than just text
- Addressing the right problem at the right time
- Targeting the right functionality
- Open and active team
- Well defined processes
- Open source reference implementations

The MPI standard covers these aspects
- Grown to be the dominant standard in HPC
- Continues evolution with MPI 4.0

The MVAPICH project has been crucial
- Open source test platform
- Quick adoption of new features
- First implementation to provide extensive data through MPI_T

But: We need to stay aware of and adjust to changing landscapes