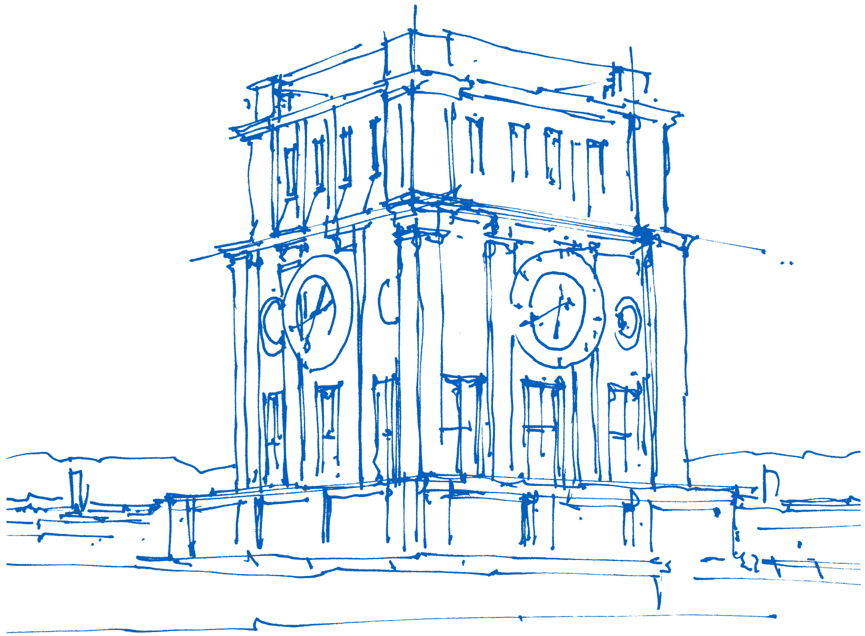
Just Writing a Standard is Not Enough!

Martin Schulz

Technical University of Munich
Department of Informatics

Chair of the MPI Forum

MVAPICH User’s Group Meeting
Columbus, Ohio, USA
August 2018



Uhrenturm der TUM

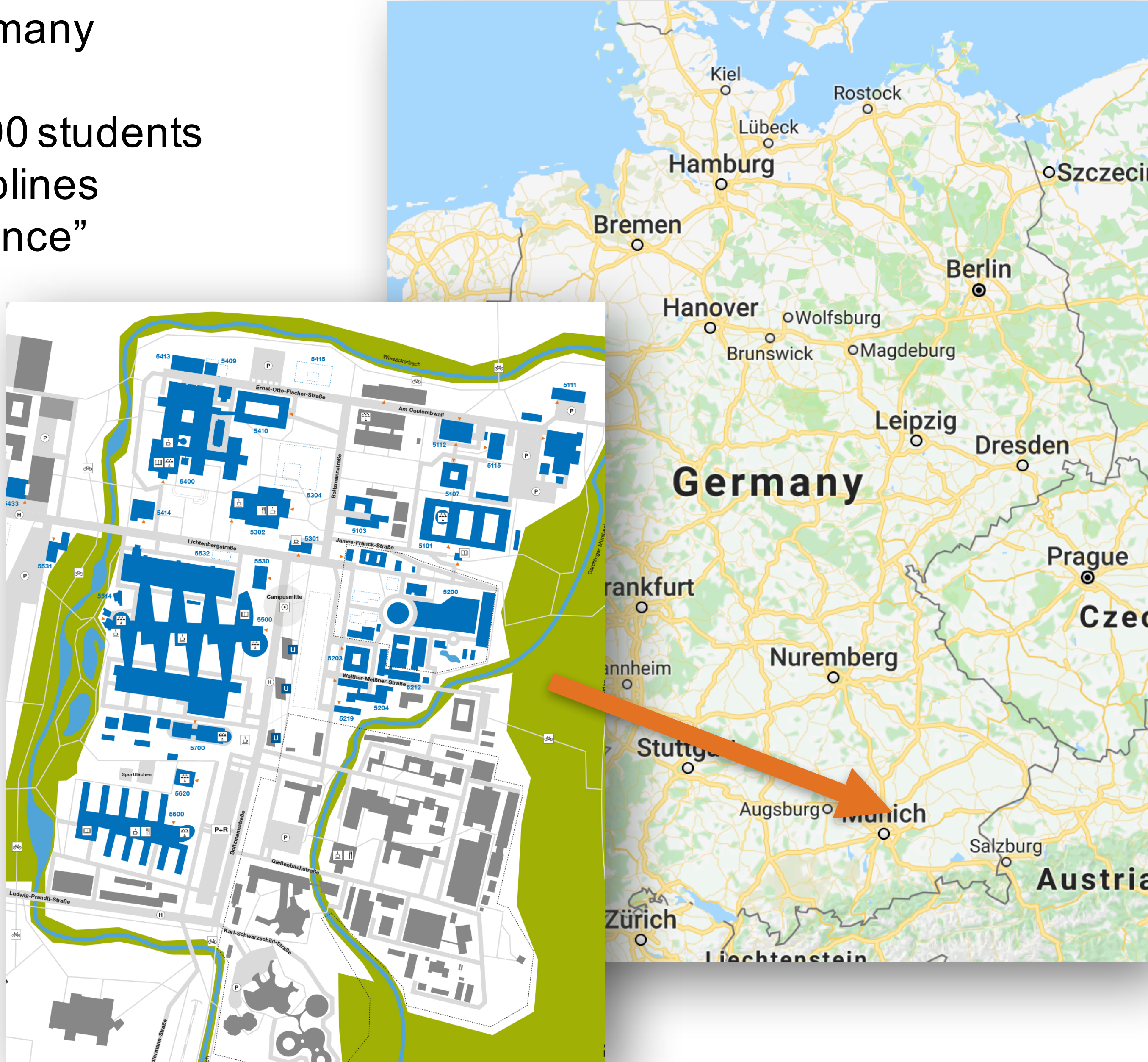
Technical University of Munich

Located in Munich, Germany

- Founded in 1868
- Around 45.000-50.000 students
- All Engineering disciplines
- “University of Excellence”

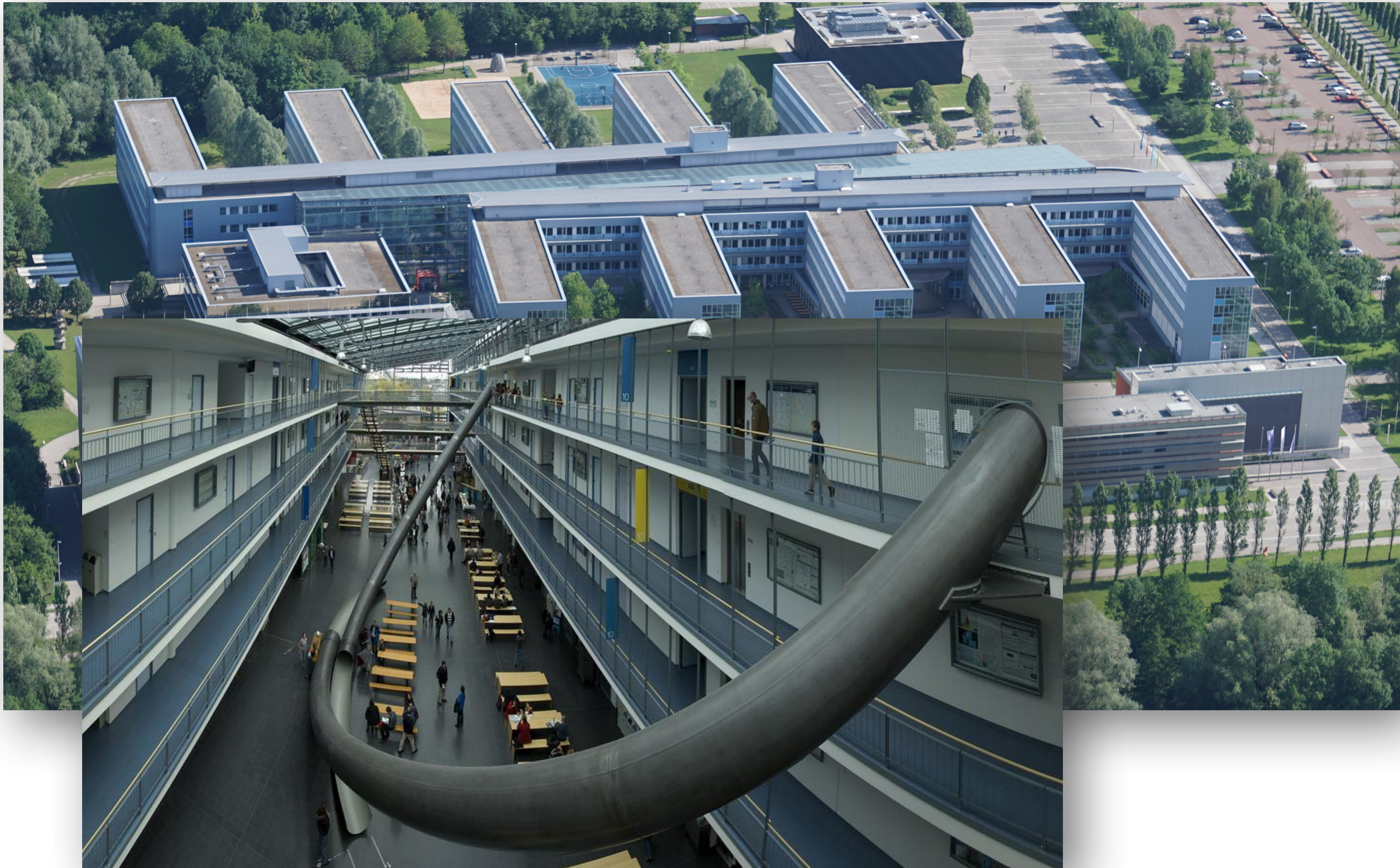
Multiple locations

- 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

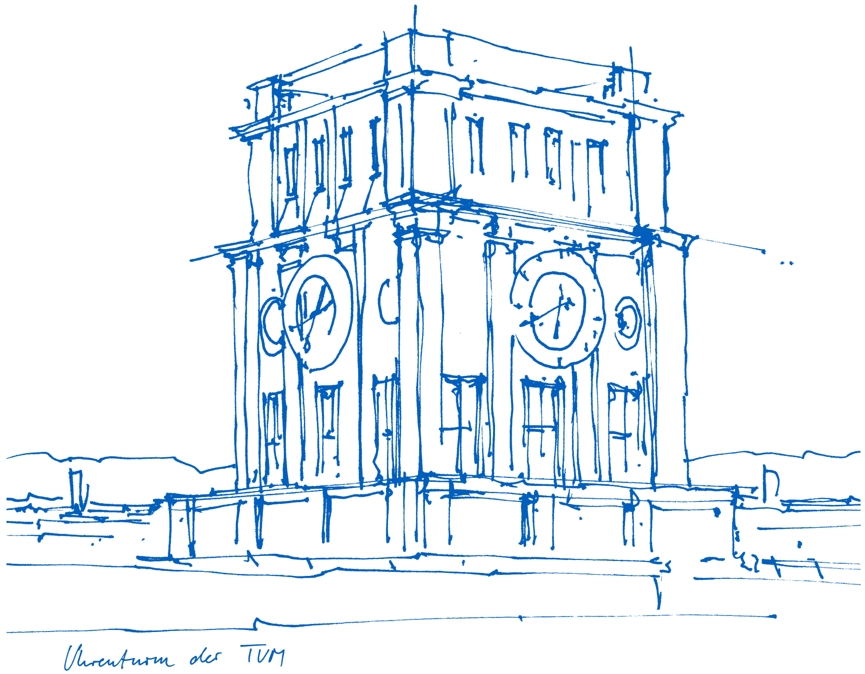
Just Writing a Standard is Not Enough!

Martin Schulz

Technical University of Munich
Department of Informatics

Chair of the MPI Forum

MVAPICH User's Group Meeting
Columbus, Ohio, USA
August 2018



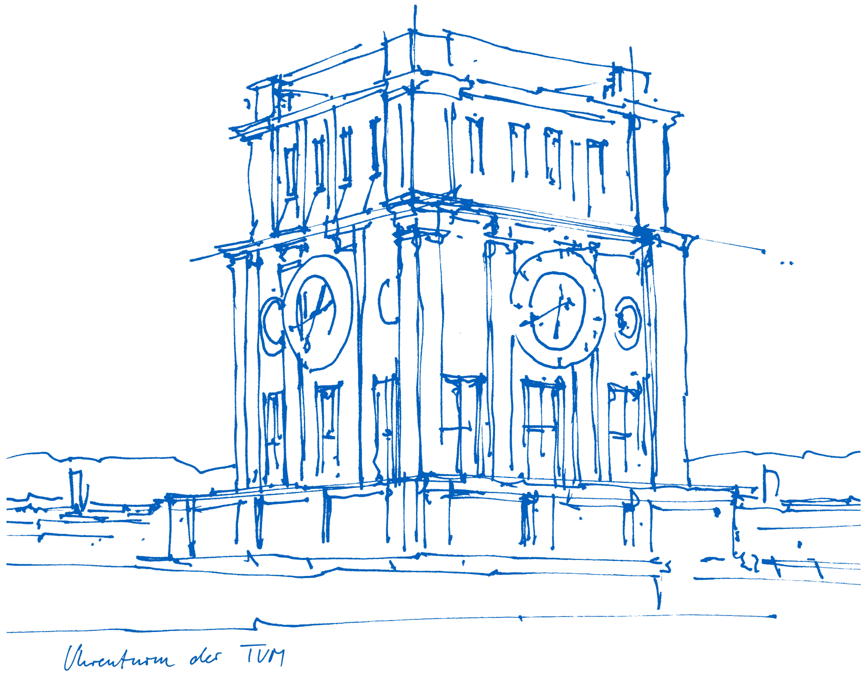
Just Writing a Standard is Not Enough!
- or –
What Made MPI What it is Today?

Martin Schulz

Technical University of Munich
Department of Informatics

Chair of the MPI Forum

MVAPICH User's Group Meeting
Columbus, Ohio, USA
August 2018

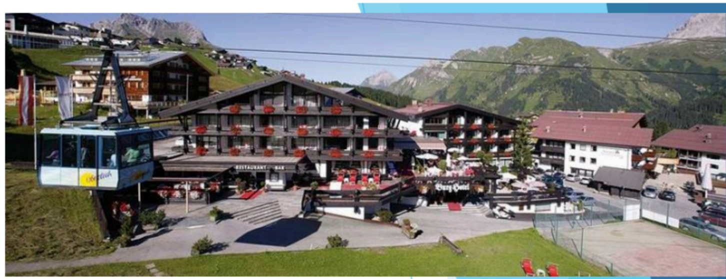


MPI is (over) 25 Years Old

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

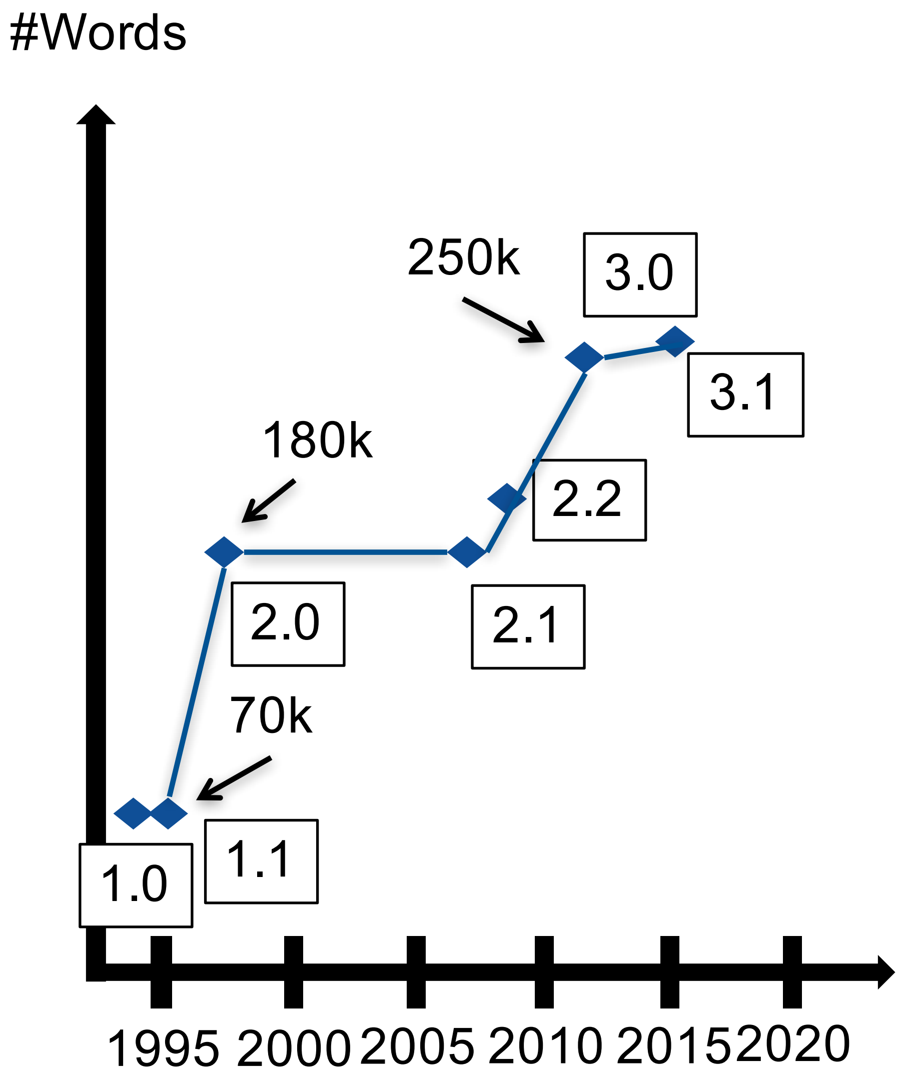
Slide from Jack Dongarra, Talk given at "Celebrating 25 Years of MPI"

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



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

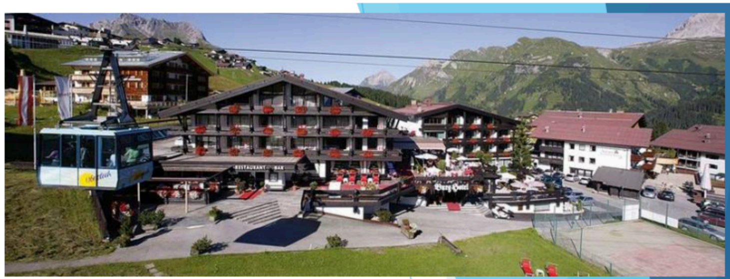
Part of many CS curricula

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

Slide from Jack Dongarra, Talk given at "Celebrating 25 Years of MPI"

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
Ambuj Singh
Amin Hassani
Amith Mamidala
Andrew Lumsdaine
Andrew Sherman
Anh Vo
Anna Rounbehler
Anne Elster
Anthony Skjellum
Antonio J. Pena
Arch Robison
Arindam Saha
Arkady Kanevsky
Arno 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 Klausecker
Christos Kavouklis
Chulho Kim
Craig Fischberg
Craig Rasmussen
Dan Nessett
Daniel Frye
Daniel Holmes
Darius Buntinas
Darren Sanders
Dave Goodell
Dave Wright
David DiNucci
David Gingold
David Goodell
David Greenberg
David Solt
David Taylor
David Walker
Davide Rossetti
Denis Nagorny
Dennis Cottel
Dennis Weeks
Devendar Bureddy
Dick Treumann
Dieter Kranzlmueller
Dmitry Durnov
Don Heller
Doug Doefler
Douglas Miller
Dries Kimpe
Ed Anderson
Ed Benson
Edgar Gabriel
Edric Ellis
Elsie Pierce
Enqiang Zhou
Erez Haba
Eric Barszcz
Eric Brunner
Eric Lantz
Eric Salo
Eric Sharakan
Erich Schikuta
Ewing Lusk
Fabian Tillier
Fred Shirley
Gabor Dozsa
Galen Shipman
Gary Howell
Geoffroy Vallee
George Bosilca
George Carr
Gil Bloch
Gilad Shainer
Greg Astfalk
Greg Bronevetsky
Greg Burns
Greg Tensa
Guillaume Mercier
Hans-Christian Hoppe
Hari Subramoni
Harish Nag
Heidi Poxon
Hideyuki Jitsumoto
Howard Palmer
Howard Pritchard
Hubert Ritzdorf
Hubertus Franke
Huiwei Lu
Huseyin S. Yildiz
Ian Foster
Ian Glendinning
Ian Stockdale
Ignacio Laguna
Jack Dongarra
James Cownie
James Dinan
James Kohl
James Pruyve
Jarek Nieplocha
Jay Lofstead
Jean-Pierre Prost
Jed Brown
Jeff Brown
Jeff Hammond
Jeff Squyres
Jeffrey Brown
Jeffrey M. Squyres
Jennifer Herrett-Skjellum
Jerome Vienne
Jerrell Watts
Jesper Larsson Traeff
Jesper Larsson Träff
Jim Cownie
Jim Feeney
Jithin Jose
Joe Baron
Joe Rieken
Joefon Jann
Joel Clark
Johann George
John Hagedorn
John Kapenga
John May
Jon Flower
Jonathan Carter
Joshua Hursey
Joshua Ladd
Juan Leon
Judith Devaney
Junchao Zhang
Kannan Narasimhan
Karl Feind
Karl Kesselman
Kathryn Mohror
Kei Harada
Keita Teranishi
Keith Underwood
Ken Raffenetti
Kento Sato
Khaled Hamidouche
Klaus Wolf
Koichi Konishi
Krishna Kandalla
Kuninobu Sasaki
Lance Shuler
Laurie Costello
Leslie Hart
Lloyd Lewins
Lyndon Clarke
Maciej Brodowicz
Manjunath Gorentla Venkata
Manojkumar Krishnan
Manuel Ujaldon
Marc Snir
Marc-Andre Hermanns
Margaret Cahir
Mario Lauria
Mark Fallon
Mark Law
Mark Pagel
Mark Sears
Martin Schulz
Marty Itzkowitz
Marydell Tholburn
Masamichi Takagi
Matthew Koop
Miao Luo
Michael Blocksome
Michael Knobloch
Min Xie
Miron Livny
Mohamad Chaarawi
Naoki Sueyasu
Nathan DeBardeleben
Nathan Doss
Nathan Hjelm
Nicholas Radcliffe
Nick M. Maclaren
Nick Nevin
Nobutoshi Sagawa
Nysal Jan
Oliver McBryan
Paddy Gillies
Pang Chen
Parkson Wong
Patrick Geoffray
Paul Pierce
Pavan Balaji
Pavel Shamis
Perry Partow
Pete Bradley
Peter Brennan
Peter Madams
Peter Ossadnik
Peter Pacheco
Peter Rigsbee
Phil McKinley
Pratap Pattnaik
Purushotham V. Bangalore
Quincey Koziol
Raffaele Giuseppe Solca
Raghunath Raja
Raghunath Raja
Chandrasekar
Rainer Keller
Raja Daoud
Rajeev Thakur
Rajesh Bordawekar
Ranier Keller
Reinhold Bader
Richard Barrett
Richard Frost
Richard L. Graham
Richard Littlefield
Richard Treumann
Rob Bjornson
Rob Ross
Robert Babb
Robert Blackmore
Robert G. Voigt
Robert George
Robert Harrison
Robert Tomlinson
Rolf Hempel
Rolf Rabenseifner
Rolf Riesen
Rolf Vandevaart
Rolf vandeVaart
Ron Brightwell
Ron Oldfield
Ryan E. Grant
Sadaf Alam
Sam Fineberg
Sameer Kumar
Sangmin Seo
Sanjay Ranka
Sayantan Sur
Scott Berryman
Scott McMillan
Shane Hebert
Shinji Sumimoto
Simon Tsang
Sreeram Potluri
Stephen Fleischman
Stephen Taylor
Stephen Wheat
Steve Hodson
Steve Kubica
Steve Landherr
Steve Otto
Steve Oyanagi
Steve Poole
Steve Zenith
Steven Huss-Lederman
Suresh Damodaran-Kamal
Susan Kraus
Susan Krauss
Swann Perarnau
Takahiro Kawashima
Takeshi Nanri
Tatsuya Abe
Terry Dontje
Terry Jones
Thom McMahon
Thomas Francois
Thomas Herault
Tim Murray
Timo Schneider
Timothy I. Mattox
Todd Gamblin
Tom Haupt
Tom Henderson
Tom Robey
Tomotake Nakamura
Tomoya Adachi
Tony Skjellum
Torsten Hoefler
Tyce McLarty
Venkat Vishwanath
Vince Fernando
Vinod Tipparaju
Weikuan Yu
Wesley Bland
William Gropp
Xin Zhao
Yann Kalemkarian
Ying Chen
Yohann Burette
Yong Cho
Yoonho Park
Yuichi Tsujita
Yutaka Ishikawa
Zhenqian Cui
Ziyang Lu

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

MPI-3.1 Impl. as of November 2015

	MPICH	MVAPICH	Open MPI	Cray MPI	Tianhe MPI	Intel MPI	IBM BG/Q MPI[1]	IBM PE MPICH[2]	IBM Platform	SGI MPI	Fujitsu MPI	MS MPI	MPC
NBC	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	(*)	Q4’15
Nbrhood collectives	✔	✔	✔	✔	✔	✔	✔	✔		✔			Q4’15
RMA	✔	✔	✔	✔	✔	✔	✔	✔		✔			*
Shared memory	✔	✔	✔	✔	✔	✔	✔	✔		✔		✔	*
Tools Interface	✔	✔	✔	✔	✔	✔	✔	✔		✔		*	Q4’16
Comm-creat group	✔	✔	✔	✔	✔	✔	✔	✔		✔			*
F08 Bindings	✔	✔	✔	✔	✔		✔			✔			Q2’16
New Datatypes	✔	✔	✔	✔	✔	✔	✔	✔		✔		✔	Q4’15
Large Counts	✔	✔	✔	✔	✔	✔	✔	✔		✔		✔	Q2’16
Matched Probe	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	✔	Q2’16
NBC I/O	✔	Q1‘16	✔	Q4‘15						Q2‘16			

[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

	MPICH	MVAPICH	Open MPI	Cray MPI	Tianhe MPI	Intel MPI	IBM BG/Q MPI[1]	IBM PE MPICH[2]	IBM Platform	SGI MPI	Fujitsu MPI	MS MPI	MPC
NBC	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	(*)	Q4’15
Nbrhood collectives	✔	✔	✔	✔	✔	✔	✔	✔		✔			Q4’15
RMA	✔	✔	✔	✔	✔	✔	✔	✔		✔			*
Shared memory	✔	✔	✔	✔	✔	✔	✔	✔		✔		✔	*
Tools Interface	✔	✔	✔	✔	✔	✔	✔	✔		✔		*	Q4’16
Comm-creat group	✔	✔	✔	✔	✔	✔	✔	✔		✔			*
F08 Bindings	✔	✔	✔	✔	✔		✔			✔			Q2’16
New Datatypes	✔	✔	✔	✔	✔	✔	✔	✔		✔		✔	Q4’15
Large Counts	✔	✔	✔	✔	✔	✔	✔	✔		✔		✔	Q2’16
Matched Probe	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	✔	Q2’16
NBC I/O	✔	Q1‘16	✔	Q4‘15						Q2‘16			

[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

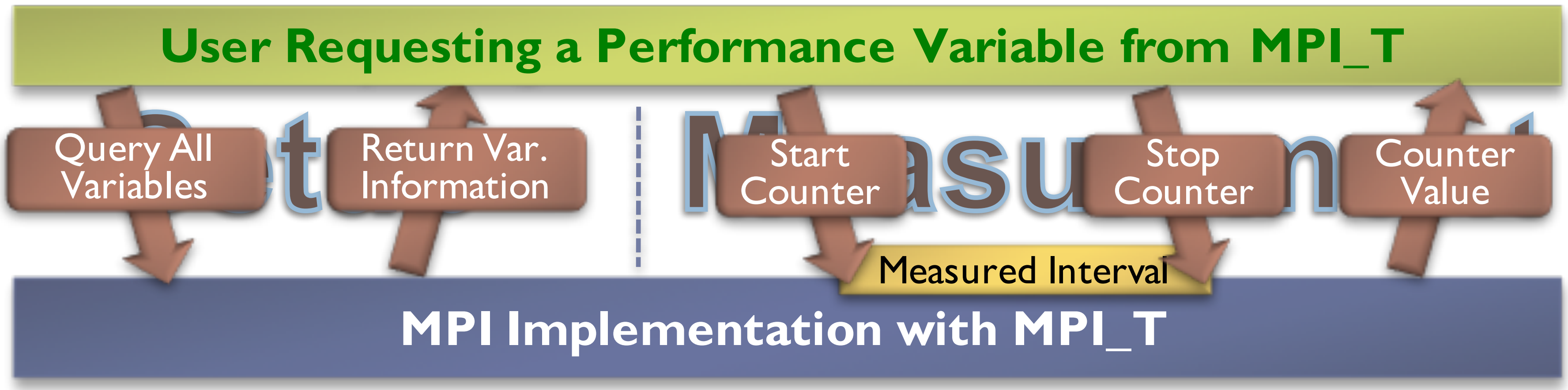
Provide access to MPI internal information
- Configuration and control information
- Performance information
- Debugging information

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



MVAPICH was Early to Provide Pvars …

2014

```
==================================
Performance Variables
==================================

Found 25 performance variables
Found 25 performance variables with verbosity <= D/A-9

Variable                                   VRB   Class   Type   Bind     R/O CNT ATM
-----------------------------------------------------------------------------------
posted_recvq_length                        U/D-2 LEVEL   UINT   n/a      YES YES  NO
unexpected_recvq_length                    U/D-2 LEVEL   UINT   n/a      YES YES  NO
posted_recvq_match_attempts                U/D-2 COUNTER UNKNOW n/a       NO YES  NO
unexpected_recvq_match_attempts            U/D-2 COUNTER UNKNOW n/a       NO YES  NO
time_failed_matching_postedq               U/D-2 TIMER   DOUBLE n/a       NO YES  NO
time_matching_unexpectedq                  U/D-2 TIMER   DOUBLE n/a       NO YES  NO
unexpected_recvq_buffer_size               U/D-2 LEVEL   UNKNOW n/a      YES YES  NO
mem_allocated                              U/B-1 LEVEL   ULLONG n/a      YES YES  NO
mem_allocated                              U/B-1 HIGHWAT ULLONG n/a      YES YES  NO
mv2_progress_poll_count                    D/B-7 COUNTER ULONG  n/a       NO  NO  NO
coll_bcast_binomial                        U/B-1 COUNTER ULLONG n/a      YES YES  NO
coll_bcast_scatter_doubling_allgather      U/B-1 COUNTER ULLONG n/a      YES YES  NO
coll_bcast_scatter_ring_allgather          U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_num_2level_comm_requests               U/D-2 COUNTER ULONG  n/a      YES YES  NO
mv2_num_2level_comm_success                U/D-2 COUNTER ULONG  n/a      YES YES  NO
mv2_num_shmem_coll_calls                   T/B-4 COUNTER ULONG  n/a      YES YES  NO
mv2_coll_bcast_binomial                    T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_doubling_allgather  T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_ring_allgather      T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_ring_allgather_shm  T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_shmem                       T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_knomial_internode           T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_knomial_intranode           T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_mcast_internode             T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_pipelined                   T/B-4 COUNTER ULLONG n/a      YES YES  NO
-----------------------------------------------------------------------------------
```

… And Continued to Expand Them

```
MPI_T Variable List
  MPI Thread support: MPI_THREAD_SINGLE
  MPI_T Thread support: MPI_THREAD_MULTIPLE

===============================
Performance Variables
===============================

Found 402 performance variables
Found 402 performance variables with verbosity <= D/A-9

Variable                                             VRB   Class   Type   Bind     R/O CNT ATM
----------------------------------------------------------------------------------------------
mem_allocated                                        U/B-1 LEVEL   ULLONG n/a      YES YES  NO
mem_allocated                                        U/B-1 HIGHWAT ULLONG n/a      YES YES  NO
num_malloc_calls                                     T/D-5 COUNTER ULONG  n/a      YES YES  NO
num_calloc_calls                                     T/D-5 COUNTER ULONG  n/a      YES YES  NO
num_memalign_calls                                   T/D-5 COUNTER ULONG  n/a      YES YES  NO
num_strdup_calls                                     T/D-5 COUNTER ULONG  n/a      YES YES  NO
num_realloc_calls                                    T/D-5 COUNTER ULONG  n/a      YES YES  NO
num_free_calls                                       T/D-5 COUNTER ULONG  n/a      YES YES  NO
num_memalign_free_calls                              T/D-5 COUNTER ULONG  n/a      YES YES  NO
mv2_num_2level_comm_requests                         U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_num_2level_comm_success                          U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_num_shmem_coll_calls                             U/B-1 COUNTER ULLONG n/a      YES YES  NO
mpit_progress_poll                                   U/B-1 COUNTER ULONG  n/a      YES YES  NO
mv2_smp_read_progress_poll                           U/B-1 COUNTER ULONG  n/a      YES YES  NO
mv2_smp_write_progress_poll                          U/B-1 COUNTER ULONG  n/a      YES YES  NO
mv2_smp_read_progress_poll_success                   U/B-1 COUNTER ULONG  n/a      YES YES  NO
mv2_smp_write_progress_poll_success                  U/B-1 COUNTER ULONG  n/a      YES YES  NO
rdma_ud_retransmissions                              U/B-1 COUNTER ULONG  n/a      YES YES  NO
mv2_coll_bcast_binomial                              U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_doubling_allgather            U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_ring_allgather                U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_ring_allgather_shm            U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_shmem                                 U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_knomial_internode                     U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_knomial_intranode                     U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_mcast_internode                       U/B-1 COUNTER ULLONG n/a      YES YES  NO
```

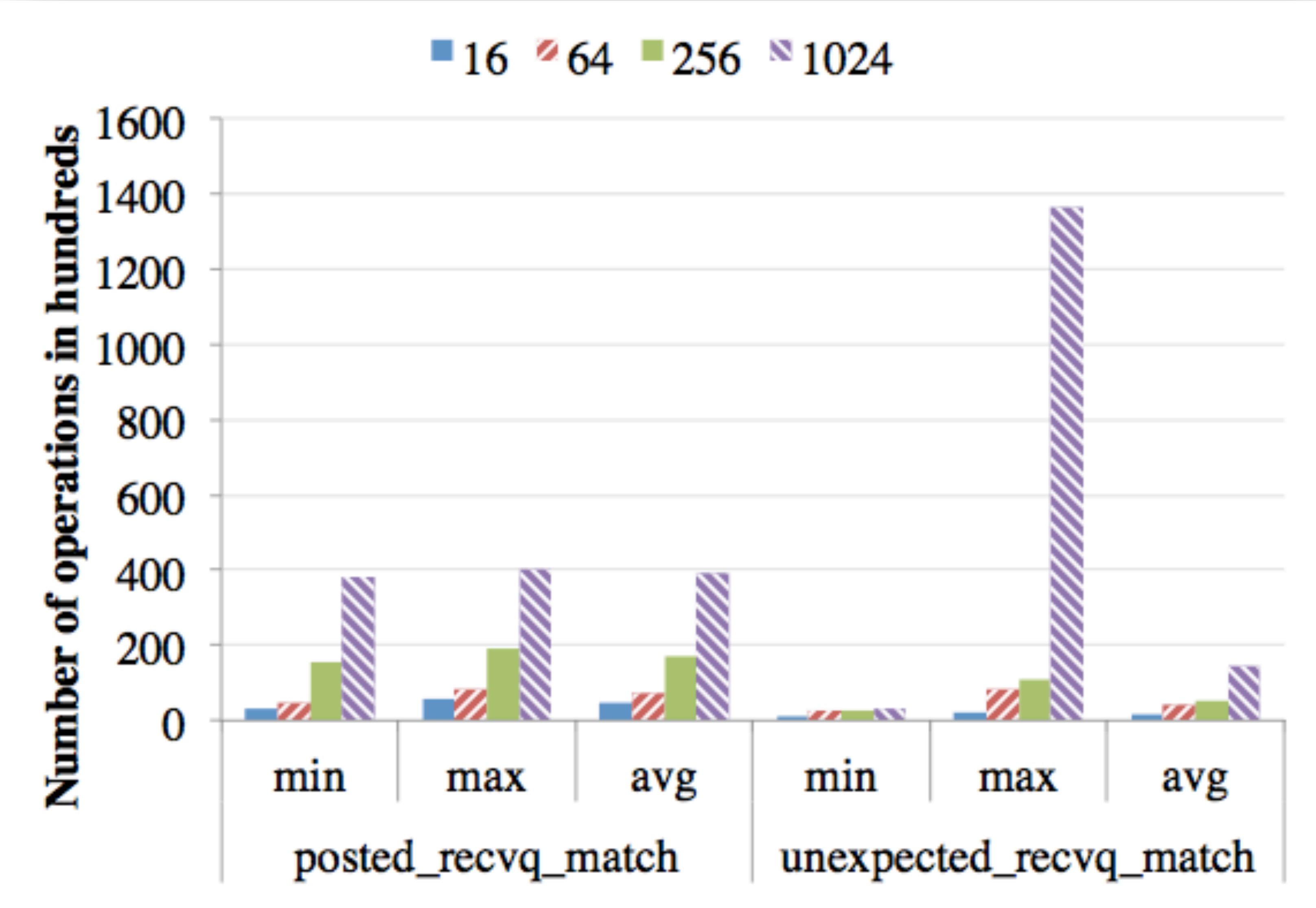
… And Continued to Expand Them

mem_allocated
mem_allocated
num_malloc_calls
num_calloc_calls
num_memalign_calls
num_strdup_calls
num_realloc_calls
num_free_calls
num_memalign_free_calls
mv2_num_2level_comm_requests
mv2_num_2level_comm_success
mv2_num_shmem_coll_calls
mpit_progress_poll
mv2_smp_read_progress_poll
mv2_smp_write_progress_poll
mv2_smp_read_progress_poll_success
mv2_smp_write_progress_poll_success
rdma_ud_retransmissions
mv2_coll_bcast_binomial
mv2_coll_bcast_scatter_doubling_allgather
mv2_coll_bcast_scatter_ring_allgather
mv2_coll_bcast_scatter_ring_allgather_shm
mv2_coll_bcast_shmem
mv2_coll_bcast_knomial_internode
mv2_coll_bcast_knomial_intranode
mv2_coll_bcast_mcast_internode
mv2_coll_bcast_pipelined
mv2_coll_bcast_binomial_bytes_send
mv2_coll_bcast_scatter_for_bcast_bytes_send
mv2_coll_bcast_scatter_doubling_allgather_bytes_send
mv2_coll_bcast_scatter_ring_allgather_bytes_send
mv2_coll_bcast_scatter_ring_allgather_shm_bytes_send
mv2_coll_bcast_knomial_internode_bytes_send
mv2_coll_bcast_knomial_intranode_bytes_send
mv2_coll_bcast_mcast_internode_bytes_send
mv2_coll_bcast_pipelined_zcpy_bytes_send
mv2_coll_bcast_tune_inter_node_helper_bytes_send
mv2_coll_bcast_inter_node_helper_bytes_send
mv2_coll_bcast_binomial_bytes_recv
mv2_coll_bcast_scatter_for_bcast_bytes_recv
mv2_coll_bcast_scatter_doubling_allgather_bytes_recv
mv2_coll_bcast_scatter_ring_allgather_bytes_recv
mv2_coll_bcast_scatter_ring_allgather_shm_bytes_recv
mv2_coll_bcast_knomial_internode_bytes_recv
mv2_coll_bcast_knomial_intranode_bytes_recv
mv2_coll_bcast_mcast_internode_bytes_recv
mv2_coll_bcast_pipelined_zcpy_bytes_recv
mv2_coll_bcast_tune_inter_node_helper_bytes_recv
mv2_coll_bcast_inter_node_helper_bytes_recv
mv2_coll_bcast_binomial_count_send
mv2_coll_bcast_scatter_for_bcast_count_send
mv2_coll_bcast_scatter_doubling_allgather_count_send
mv2_coll_bcast_scatter_ring_allgather_count_send
mv2_coll_bcast_scatter_ring_allgather_shm_count_send
mv2_coll_bcast_knomial_internode_count_send
mv2_coll_bcast_knomial_intranode_count_send
mv2_coll_bcast_mcast_internode_count_send
mv2_coll_bcast_pipelined_zcpy_count_send
mv2_coll_bcast_tune_inter_node_helper_count_send
mv2_coll_bcast_inter_node_helper_count_send
mv2_coll_bcast_binomial_count_recv
mv2_coll_bcast_scatter_for_bcast_count_recv
mv2_coll_bcast_scatter_doubling_allgather_count_recv
mv2_coll_bcast_scatter_ring_allgather_count_recv
mv2_coll_bcast_scatter_ring_allgather_shm_count_recv
mv2_coll_bcast_knomial_internode_count_recv
mv2_coll_bcast_knomial_intranode_count_recv
mv2_coll_bcast_mcast_internode_count_recv
mv2_coll_bcast_pipelined_zcpy_count_recv
mv2_coll_bcast_tune_inter_node_helper_count_recv
mv2_coll_bcast_inter_node_helper_count_recv
mv2_coll_bcast_bytes_send
mv2_coll_bcast_bytes_recv
mv2_coll_bcast_count_send
mv2_coll_bcast_count_recv
mv2_coll_alltoall_inplace
mv2_coll_alltoall_bruck
mv2_coll_alltoall_rd
mv2_coll_alltoall_sd
mv2_coll_alltoall_pw
mv2_coll_alltoall_inplace_bytes_send
mv2_coll_alltoall_bruck_bytes_send
mv2_coll_alltoall_sd_bytes_send
mv2_coll_alltoall_pw_bytes_send
mv2_coll_alltoall_inplace_bytes_recv
mv2_coll_alltoall_bruck_bytes_recv
mv2_coll_alltoall_sd_bytes_recv
mv2_coll_alltoall_pw_bytes_recv
mv2_coll_alltoall_inplace_count_send
mv2_coll_alltoall_bruck_count_send
mv2_coll_alltoall_sd_count_send
mv2_coll_alltoall_pw_count_send
mv2_coll_alltoall_inplace_count_recv
mv2_coll_alltoall_bruck_count_recv
mv2_coll_alltoall_sd_count_recv
mv2_coll_alltoall_pw_count_recv
mv2_coll_alltoall_bytes_send
mv2_coll_alltoall_bytes_recv
mv2_coll_alltoall_count_send
mv2_coll_alltoall_count_recv
mv2_coll_alltoall_intra_bytes_send
mv2_coll_alltoall_intra_bytes_recv
mv2_coll_alltoall_intra_count_send
mv2_coll_alltoall_intra_count_recv
mv2_coll_alltoallv_pw
mv2_coll_alltoallv_intra_bytes_send
mv2_coll_alltoallv_intra_bytes_recv
mv2_coll_alltoallv_intra_count_send
mv2_coll_alltoallv_intra_count_recv
mv2_coll_alltoallv_bytes_send
mv2_coll_alltoallv_bytes_recv
mv2_coll_alltoallv_count_send
mv2_coll_alltoallv_count_recv
mv2_coll_alltoall_cuda_intra_bytes_send
mv2_coll_alltoall_cuda_intra_bytes_recv
mv2_coll_alltoall_cuda_intra_count_send
mv2_coll_alltoall_cuda_intra_count_recv
mv2_coll_alltoall_cuda_bytes_send
mv2_coll_alltoall_cuda_bytes_recv
mv2_coll_alltoall_cuda_count_send
mv2_coll_alltoall_cuda_count_recv
mv2_coll_allreduce_sharp
mv2_coll_allreduce_shm_rd
mv2_coll_allreduce_shm_rs
mv2_coll_allreduce_shm_intra
mv2_coll_allreduce_intra_p2p
mv2_coll_allreduce_2lvl
mv2_coll_allreduce_shmem
mv2_coll_allreduce_mcast
mv2_coll_allreduce_reduce_scatter_allgather_colls
mv2_coll_allreduce_pt2pt_rd_bytes_send
mv2_coll_allreduce_pt2pt_rs_bytes_send
mv2_coll_allreduce_pt2pt_rd_bytes_recv
mv2_coll_allreduce_pt2pt_rs_bytes_recv
mv2_coll_allreduce_pt2pt_rd_count_send
mv2_coll_allreduce_pt2pt_rs_count_send
mv2_coll_allreduce_pt2pt_rd_count_recv
mv2_coll_allreduce_pt2pt_rs_count_recv
mv2_coll_allreduce_bytes_send
mv2_coll_allreduce_bytes_recv
mv2_coll_allreduce_count_send
mv2_coll_allreduce_count_recv
mv2_coll_allgather_rd_allgather_comm
mv2_coll_allgather_rd
mv2_coll_allgather_bruck
mv2_coll_allgather_ring
mv2_coll_allgather_direct
mv2_coll_allgather_directspread
mv2_coll_allgather_gather_bcast
mv2_coll_allgather_2lvl_nonblocked
mv2_coll_allgather_2lvl_ring_nonblocked
mv2_coll_allgather_2lvl_direct
mv2_coll_allgather_2lvl_ring
mv2_coll_allgather_rd_bytes_send
mv2_coll_allgather_bruck_bytes_send
mv2_coll_allgather_ring_bytes_send
mv2_coll_allgather_direct_bytes_send
mv2_coll_allgather_directspread_bytes_send
mv2_coll_allgather_2lvl_ring_nonblocked_bytes_send
mv2_coll_allgather_2lvl_direct_bytes_send
mv2_coll_allgather_2lvl_ring_bytes_send
mv2_coll_allgather_rd_bytes_recv
mv2_coll_allgather_bruck_bytes_recv
mv2_coll_allgather_ring_bytes_recv
mv2_coll_allgather_direct_bytes_recv
mv2_coll_allgather_directspread_bytes_recv
mv2_coll_allgather_2lvl_ring_nonblocked_bytes_recv
mv2_coll_allgather_2lvl_direct_bytes_recv
mv2_coll_allgather_2lvl_ring_bytes_recv
mv2_coll_allgather_rd_count_send
mv2_coll_allgather_bruck_count_send
mv2_coll_allgather_ring_count_send
mv2_coll_allgather_direct_count_send
mv2_coll_allgather_directspread_count_send
mv2_coll_allgather_2lvl_ring_nonblocked_count_send
mv2_coll_allgather_2lvl_direct_count_send
mv2_coll_allgather_2lvl_ring_count_send
mv2_coll_allgather_rd_count_recv
mv2_coll_allgather_bruck_count_recv
mv2_coll_allgather_ring_count_recv
mv2_coll_allgather_direct_count_recv
mv2_coll_allgather_directspread_count_recv
mv2_coll_allgather_2lvl_ring_nonblocked_count_recv
mv2_coll_allgather_2lvl_direct_count_recv
mv2_coll_allgather_2lvl_ring_count_recv
mv2_coll_allgather_bytes_send
mv2_coll_allgather_bytes_recv
mv2_coll_allgather_count_send
mv2_coll_allgather_count_recv
mv2_coll_allgather_cuda_intra_bytes_send
mv2_coll_allgather_cuda_intra_bytes_recv
mv2_coll_allgather_cuda_intra_count_send
mv2_coll_allgather_cuda_intra_count_recv
mv2_coll_allgather_cuda_bytes_send
mv2_coll_allgather_cuda_bytes_recv
mv2_coll_allgather_cuda_count_send
mv2_coll_allgather_cuda_count_recv
mv2_coll_gather_pt2pt
mv2_coll_gather_direct
mv2_coll_gather_direct_blk
mv2_coll_gather_two_level_direct
mv2_coll_gather_limic_scheme_pt_pt
mv2_coll_gather_limic_scheme_pt_linear
mv2_coll_gather_limic_scheme_linear_pt
mv2_coll_gather_limic_scheme_linear_linear
mv2_coll_gather_limic_scheme_single_leader
mv2_coll_gather_intra_node_limic
mv2_coll_gather_direct_bytes_send
mv2_coll_gather_direct_blk_bytes_send
mv2_coll_gather_two_level_direct_bytes_send
mv2_coll_gather_direct_bytes_recv
mv2_coll_gather_direct_blk_bytes_recv
mv2_coll_gather_two_level_direct_bytes_recv
mv2_coll_gather_direct_count_send
mv2_coll_gather_direct_blk_count_send
mv2_coll_gather_two_level_direct_count_send
mv2_coll_gather_direct_count_recv
mv2_coll_gather_direct_blk_count_recv
mv2_coll_gather_two_level_direct_count_recv
mv2_coll_gather_bytes_send
mv2_coll_gather_bytes_recv
mv2_coll_gather_count_send
mv2_coll_gather_count_recv
mv2_coll_reduce_scatter_basic
mv2_coll_reduce_scatter_rec_halving
mv2_coll_reduce_scatter_pairwise
mv2_coll_reduce_scatter_non_comm
mv2_coll_reduce_scatter_noncomm
mv2_coll_reduce_scatter_noncomm_bytes_send
mv2_coll_reduce_scatter_basic_bytes_send
mv2_coll_reduce_scatter_rec_halving_bytes_send
mv2_coll_reduce_scatter_pairwise_bytes_send
mv2_coll_reduce_scatter_non_comm_bytes_send
mv2_coll_reduce_scatter_noncomm_bytes_recv
mv2_coll_reduce_scatter_basic_bytes_recv
mv2_coll_reduce_scatter_rec_halving_bytes_recv
mv2_coll_reduce_scatter_pairwise_bytes_recv
mv2_coll_reduce_scatter_non_comm_bytes_recv
mv2_coll_reduce_scatter_noncomm_count_send
mv2_coll_reduce_scatter_basic_count_send
mv2_coll_reduce_scatter_rec_halving_count_send
mv2_coll_reduce_scatter_pairwise_count_send
mv2_coll_reduce_scatter_non_comm_count_send
mv2_coll_reduce_scatter_noncomm_count_recv
mv2_coll_reduce_scatter_basic_count_recv
mv2_coll_reduce_scatter_rec_halving_count_recv
mv2_coll_reduce_scatter_pairwise_count_recv
mv2_coll_reduce_scatter_non_comm_count_recv
mv2_coll_reduce_scatter_bytes_send
mv2_coll_reduce_scatter_bytes_recv
mv2_coll_reduce_scatter_count_send
mv2_coll_reduce_scatter_count_recv
mv2_coll_scatter_mcast
mv2_coll_scatter_binomial
mv2_coll_scatter_direct
mv2_coll_scatter_direct_blk
mv2_coll_scatter_two_level_binomial
mv2_coll_scatter_two_level_direct
mv2_coll_scatter_inter
mv2_coll_scatter_mcast_bytes_recv
mv2_coll_scatter_mcast_bytes_send
mv2_coll_scatter_binomial_bytes_recv
mv2_coll_scatter_binomial_bytes_send
mv2_coll_scatter_direct_bytes_recv
mv2_coll_scatter_direct_bytes_send
mv2_coll_scatter_direct_blk_bytes_recv
mv2_coll_scatter_direct_blk_bytes_send
mv2_coll_scatter_two_level_direct_bytes_recv
mv2_coll_scatter_two_level_direct_bytes_send
mv2_coll_scatter_two_level_binomial_bytes_recv
mv2_coll_scatter_two_level_binomial_bytes_send
mv2_coll_scatter_inter_bytes_recv
mv2_coll_scatter_inter_bytes_send
mv2_coll_scatter_mcast_count_recv
mv2_coll_scatter_mcast_count_send
mv2_coll_scatter_binomial_count_recv
mv2_coll_scatter_binomial_count_send
mv2_coll_scatter_direct_count_recv
mv2_coll_scatter_direct_count_send
mv2_coll_scatter_direct_blk_count_recv
mv2_coll_scatter_direct_blk_count_send
mv2_coll_scatter_two_level_direct_count_recv
mv2_coll_scatter_two_level_direct_count_send
mv2_coll_scatter_two_level_binomial_count_recv
mv2_coll_scatter_two_level_binomial_count_send
mv2_coll_scatter_inter_count_recv
mv2_coll_scatter_inter_count_send
mv2_coll_scatter_bytes_send
mv2_coll_scatter_bytes_recv
mv2_coll_scatter_count_send
mv2_coll_scatter_count_recv
mv2_coll_reduce_binomial
mv2_coll_reduce_redscat_gather
mv2_coll_reduce_shmem
mv2_coll_reduce_knomial
mv2_coll_reduce_zcpy
mv2_coll_reduce_binomial_bytes_send
mv2_coll_reduce_redscat_gather_bytes_send
mv2_coll_reduce_two_level_helper_bytes_send
mv2_coll_reduce_knomial_bytes_send
mv2_coll_reduce_zcpy_bytes_send
mv2_coll_reduce_binomial_bytes_recv
mv2_coll_reduce_redscat_gather_bytes_recv
mv2_coll_reduce_two_level_helper_bytes_recv
mv2_coll_reduce_knomial_bytes_recv
mv2_coll_reduce_zcpy_bytes_recv
mv2_coll_reduce_binomial_count_send
mv2_coll_reduce_redscat_gather_count_send
mv2_coll_reduce_two_level_helper_count_send
mv2_coll_reduce_knomial_count_send
mv2_coll_reduce_zcpy_count_send
mv2_coll_reduce_binomial_count_recv
mv2_coll_reduce_redscat_gather_count_recv
mv2_coll_reduce_two_level_helper_count_recv
mv2_coll_reduce_knomial_count_recv
mv2_coll_reduce_zcpy_count_recv
mv2_coll_reduce_bytes_send
mv2_coll_reduce_bytes_recv
mv2_coll_reduce_count_send
mv2_coll_reduce_count_recv
mv2_coll_gatherv_algo
mv2_coll_gatherv_default_bytes_send
mv2_coll_gatherv_default_bytes_recv
mv2_coll_gatherv_default_count_send
mv2_coll_gatherv_default_count_recv
mv2_coll_gatherv_bytes_send
mv2_coll_gatherv_bytes_recv
mv2_coll_gatherv_count_send
mv2_coll_gatherv_count_recv
mv2_coll_allgatherv_rec_doubling
mv2_coll_allgatherv_bruck
mv2_coll_allgatherv_ring
mv2_coll_allgatherv_ring_cyclic
mv2_coll_barrier_pairwise
mv2_coll_barrier_shmem
mv2_coll_barrier_pairwise_bytes_send
mv2_coll_barrier_pairwise_bytes_recv
mv2_coll_barrier_pairwise_count_send
mv2_coll_barrier_pairwise_count_recv
mv2_coll_barrier_bytes_send
mv2_coll_barrier_bytes_recv
mv2_coll_barrier_count_send
mv2_coll_barrier_count_recv
mv2_coll_allgatherv_rec_doubling_bytes_send
mv2_coll_allgatherv_bruck_bytes_send
mv2_coll_allgatherv_ring_bytes_send
mv2_coll_allgatherv_rec_doubling_bytes_recv
mv2_coll_allgatherv_bruck_bytes_recv
mv2_coll_allgatherv_ring_bytes_recv
mv2_coll_allgatherv_rec_doubling_count_send
mv2_coll_allgatherv_bruck_count_send
mv2_coll_allgatherv_ring_count_send
mv2_coll_allgatherv_rec_doubling_count_recv
mv2_coll_allgatherv_bruck_count_recv
mv2_coll_allgatherv_ring_count_recv
mv2_coll_allgatherv_bytes_send
mv2_coll_allgatherv_bytes_recv
mv2_coll_allgatherv_count_send
mv2_coll_allgatherv_count_recv
mv2_coll_exscan_algo
mv2_coll_scan_algo
mv2_reg_cache_hits
mv2_reg_cache_misses
mv2_vbuf_allocated
mv2_vbuf_freed
mv2_ud_vbuf_allocated
mv2_ud_vbuf_freed
mv2_vbuf_available
mv2_smp_eager_avail_buffer
mv2_smp_rndv_avail_buffer
mv2_smp_eager_total_buffer
mv2_smp_rndv_total_buffer
mv2_smp_eager_buffer_max_use
mv2_smp_rndv_buffer_max_use
mv2_ud_vbuf_available
mv2_smp_eager_sent
mv2_smp_rndv_sent
mv2_smp_eager_received
mv2_smp_rndv_received
mv2_rdmafp_ctrl_packet_count
mv2_rdmafp_out_of_order_packet_count
mv2_rdmafp_exact_recv_count
mv2_ibv_channel_ctrl_packet_count
mv2_ibv_channel_out_of_order_packet_count
mv2_ibv_channel_exact_recv_count
mv2_coll_iscatter_binomial_bytes_send
mv2_coll_iscatter_binomial_bytes_recv
mv2_coll_iscatter_binomial_count_send
mv2_coll_iscatter_binomial_count_recv
mv2_coll_iscatter_bytes_send
mv2_coll_iscatter_bytes_recv
mv2_coll_iscatter_count_send
mv2_coll_iscatter_count_recv
posted_recvq_length
unexpected_recvq_length
posted_recvq_match_attempts
unexpected_recvq_match_attempts
time_failed_matching_postedq
time_failed_matching_postedq
time_matching_unexpectedq
time_matching_unexpectedq
unexpected_recvq_buffer_size

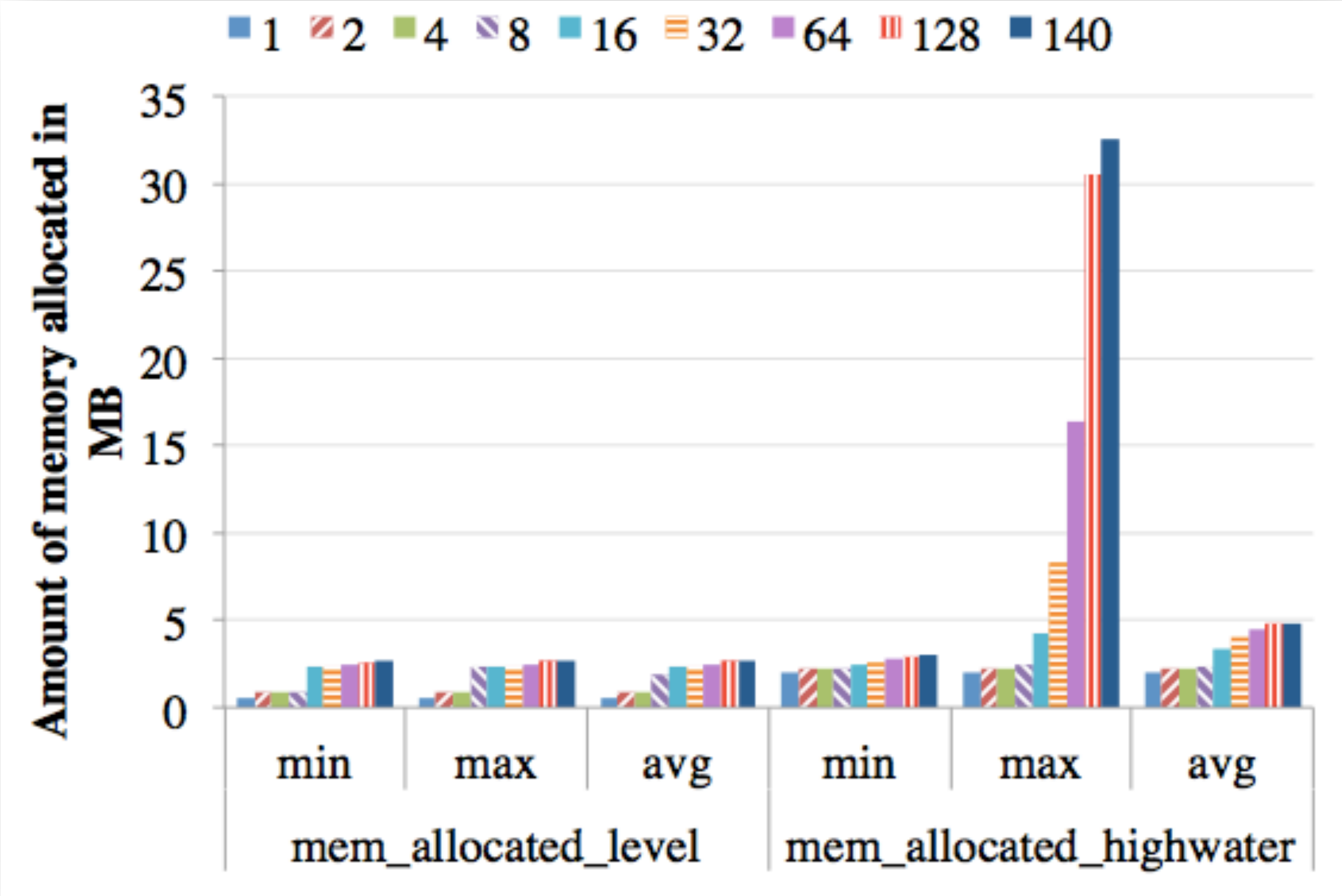
Examples of Counters

Variable	Description
posted_recvq_match	Counts how many times the queue for receiving expected messages is read.
unexpected_recvq_match	Counts how many times the queue for receiving unexpected messages is read.
progress_poll_count	Counts how many times the application polls the progress of a communication. The higher the value, the more CPU time is spent in polling.
mem_allocated_level	Gives the instantaneous memory usage by the library in bytes.
mem_allocated_highwater	Gives the maximum number of bytes ever allocated by the MPI library at a given process for the duration of the application.
coll_bcast_binom	Counts how many of the MPI broadcast collective calls use the Binomial algorithm during an application run.
num_shmem_coll	Counts how many of the collective communication calls are using shared memory.
coll_bcast_shmem	Counts how many of the MPI broadcast communication calls are shared memory based collectives.

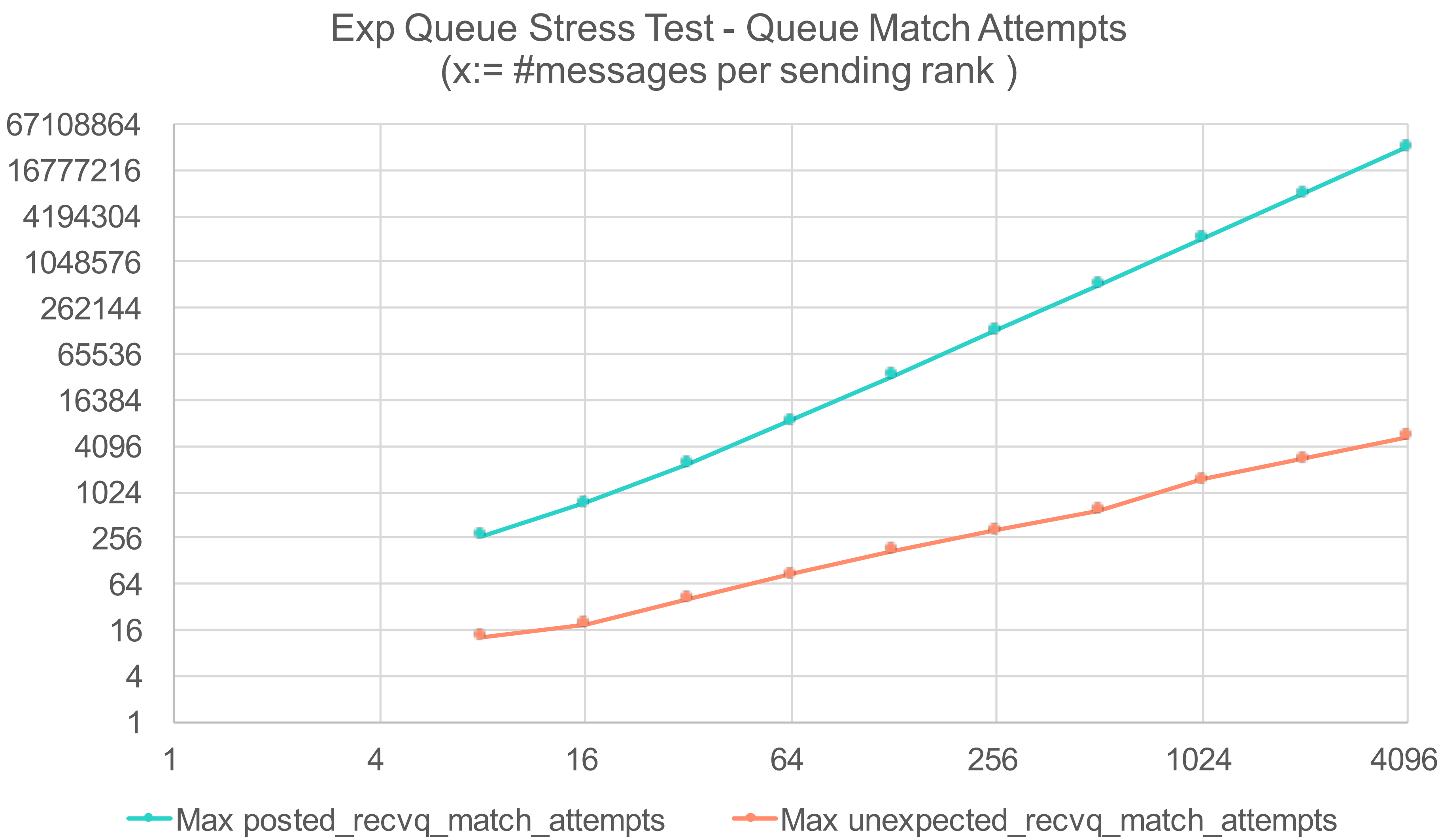
Example: Receive Queues for NAS BT



Example: Memory Consumptions for NEK5000



Example: Validation of Message Queue Stresstest



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