“Abstract is good, but ... a bit much like a technical talk?”

Thanks for organizing this!
My personal journey with MPI

Disposable income distribution

Chemnitz University of Technology
Diploma Thesis
Evaluation of publicly available Barrier-Algorithms and Improvement of the Barrier-Operation for large-scale Cluster-Systems with special Attention on InfiniBand Networks

Torsten Höffer
hos@informatik.tu-chemnitz.de

Advisors: Dipl.-Inf. T. Mehlau, Dipl.-Inf. F. Motte
Supervisor: Prof. Dr.-Ing. W. Rehm

<17k <20k <23k <26k <29k <33k
Nonblocking collective operations – first discussed in MPI-1!

- `MPI_I<collective>(args, MPI_Request *req);`

Implementation and performance analysis of non-blocking collective operations for MPI. Abstract: ... LibNBC provides non-blocking versions of all MPI collective operations, is layered on top of MPI-1, and is portable to nearly all parallel architectures.

EuroPVM/MPI’06 Speedup for Jacobi/CG

![Graphs showing speedup for different network interconnects: Gigabit Ethernet and InfiniBand.](image)

Fig. 3. Parallel speedup for different network interconnects.

Message progression in parallel computing - to thread or not to thread ...

![Image of a book](image)

six years later
ten years later
But wait, nonblocking barriers, seriously?

... turns out to be very useful after all:

Scalable communication protocols for dynamic sparse data exchange
dL.acm.org/citation.cfm?id=1693476
by T Hoeffler - 2010 - Cited by 41 - Related articles
Jan 9, 2010 - We define the dynamic sparse data-exchange (DSDE) problem and derive bounds in the well known LogGP model. While current approaches ...
Neighborhood Collectives

- Just datatypes for collectives – default collectives are “contiguous”, neighbor collectives are user-defined

One common misconception with MPI datatypes is that they are slow. Early in the life of MPI, using MPI datatypes to achieve communication was often slower than packing the data by hand. Datatype performance has been and continues to be an active area of research, allowing datatype implementations to achieve much higher performance. Some MPI implementations are even capable of doing scatter/gather sends and receives, completely eliminating the need to pack messages for transfer. In short, poor datatype performance is generally a thing of the past, and it's getting better every day.

We need to focus on optimizing MPI-3 now (similar issues for RMA etc.)
State of MPI today – programming has changed dramatically

until 10 years ago

today’s programming

And the domain scientists?

OpenMP

OpenACC

NVIDIA CUDA

OpenCL

MPI
HPC community codes towards the end of Moore’s law (i.e., age of acceleration)

'07: Fortran + MPI

'12: Fortran + MPI + C++ (DSL) + CUDA

'13: Fortran + MPI + C++ (DSL) + CUDA + OpenACC

'???: Fortran + MPI + C++ (DSL) + CUDA + OpenACC + XXX

What is with the MPI community and how can we help?
MPI’s own Innovator’s Dilemma

- We should have a bold research strategy to go forward!

Replace MPI?

Rethink MPI!

Distributed CUDA
- Run MPI right on your GPU (SC’16)

streaming Processing in Network
- CUDA for Network Cards (SC’17)

MPI for Big Data
- Distributed Join Algorithms on Thousands of Cores (VLDB’17)
Let’s move MPI to new heights!

https://spcl.inf.ethz.ch/Jobs/