MPI vs the Commercialization of HPC

Ideas for a modern MPI

Joseph Schuchart

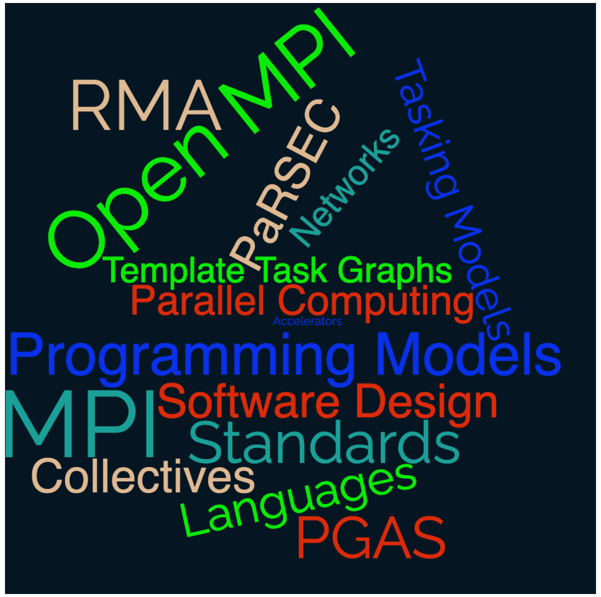
EuroMPI’23, September 16, Bristol, UK

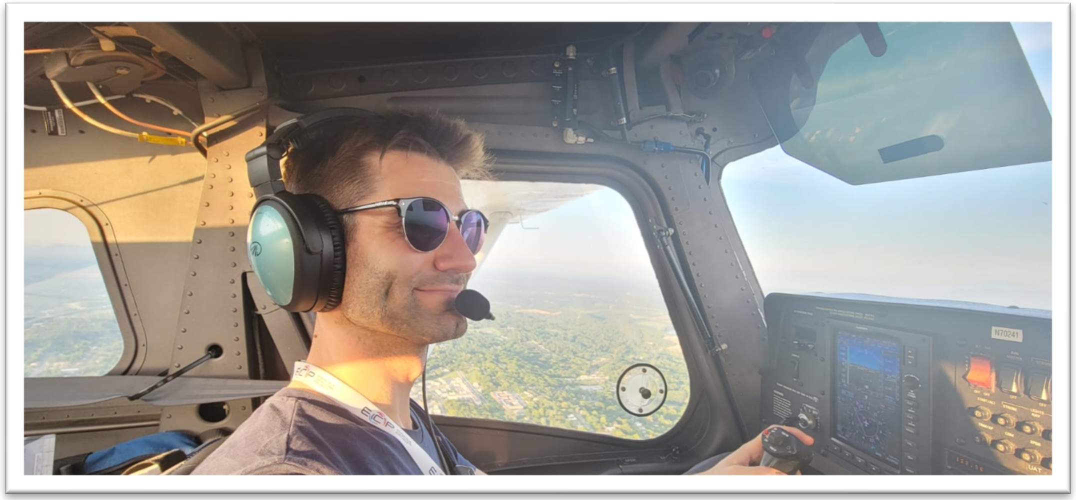
ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

About Me

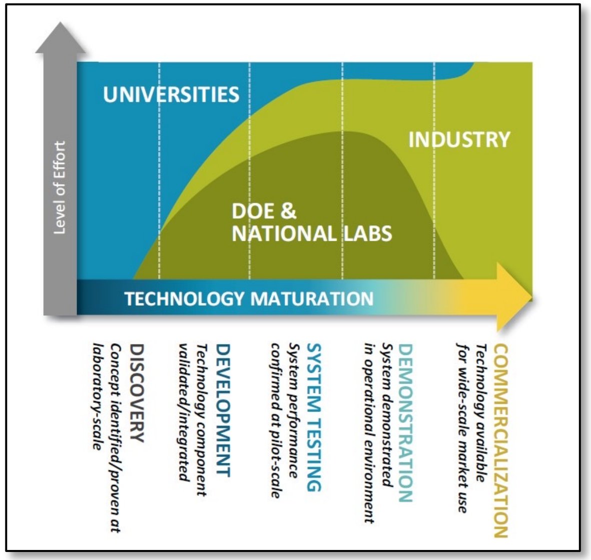
- Masters: TU Dresden (2012)
- PhD: Stuttgart University (2020)
- Research Scientist @ ICL
- First Forum Meeting: May 2019





The HPC Landscape Today

- Commercialization of HPC
- Accelerators (GPU, APU, TPU, Quantum?)
- Alternative communication libraries (NCCL, RCCL)
- Decline in public funding


Level of Effort

UNIVERSITIES

INDUSTRY

DOE & NATIONAL LABS

TECHNOLOGY MATURATION

DISCOVERY
Concept identified/proven at laboratory-scale

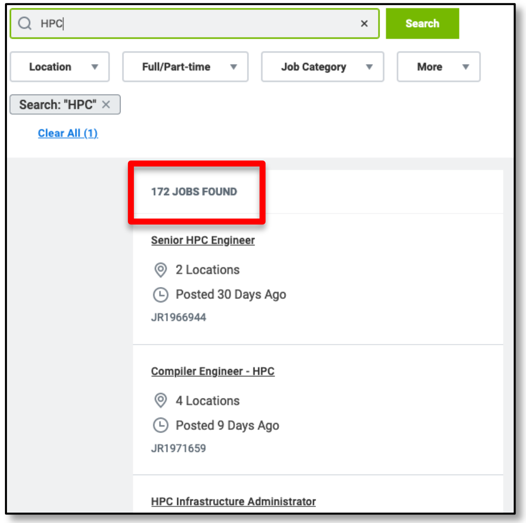
DEVELOPMENT
Technology component validated/integrated

SYSTEM TESTING
System performance confirmed at pilot-scale

DEMONSTRATION
System demonstrated in operational environment

COMMERCIALIZATION
Technology available for wide-scale market use

https://netl.doe.gov/business/partnerships



HPC

Search: "HPC" ×

Clear All (1)

172 JOBS FOUND

Senior HPC Engineer

2 Locations

Posted 30 Days Ago

JR1966944

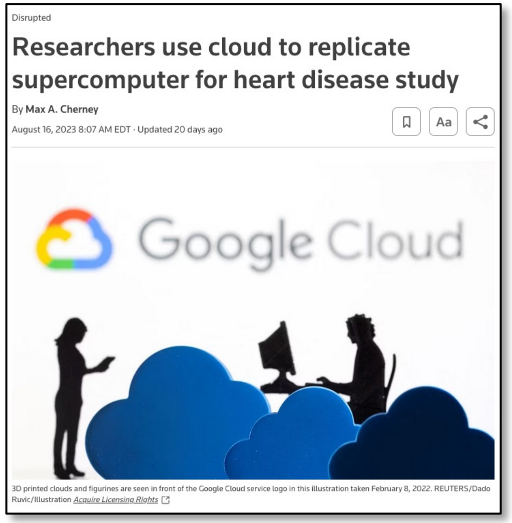
Compiler Engineer - HPC

4 Locations

Posted 9 Days Ago

JR1971659

HPC Infrastructure Administrator



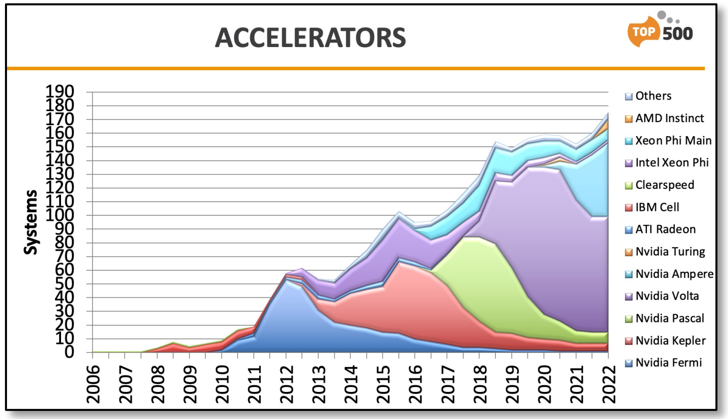
Disrupted

Researchers use cloud to replicate supercomputer for heart disease study

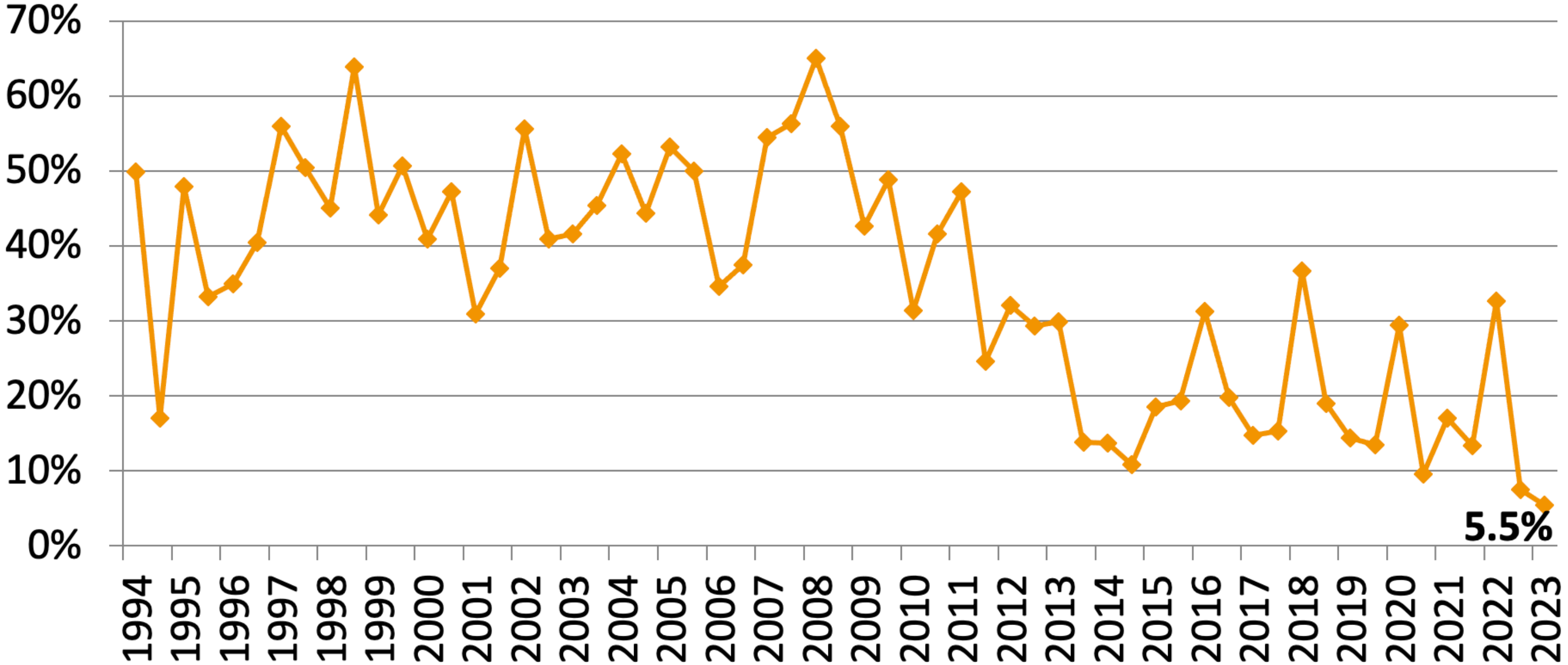
By **Max A. Cherney**

August 16, 2023 8:07 AM EDT · Updated 20 days ago

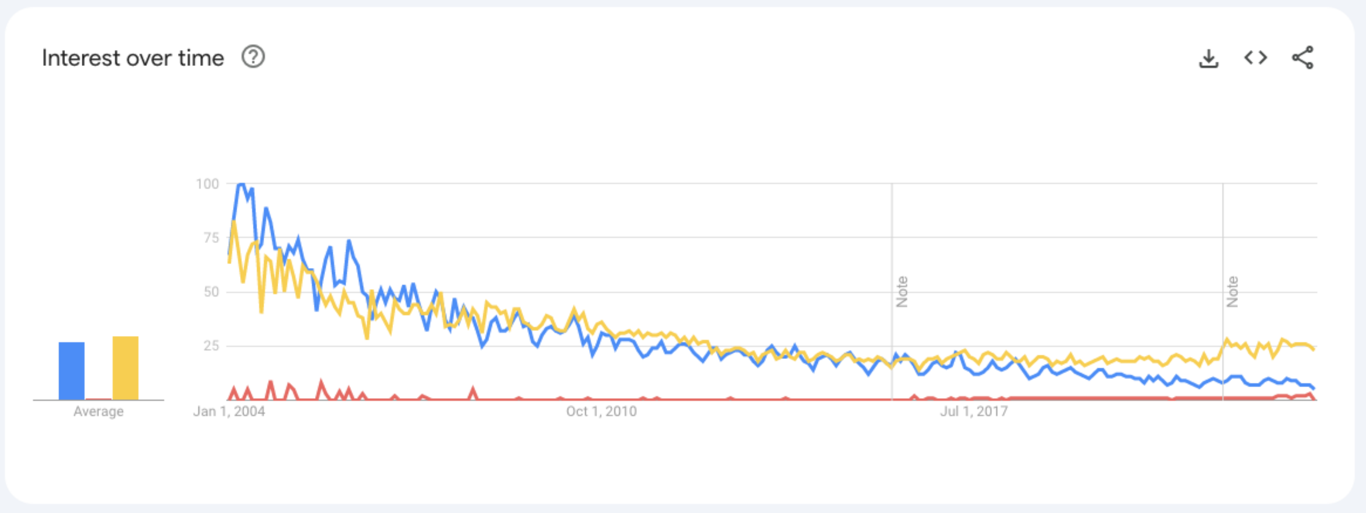
3D printed clouds and figurines are seen in front of the Google Cloud service logo in this illustration taken February 8, 2022. REUTERS/Dado Ruvic/Illustration *Acquire Licensing Rights*

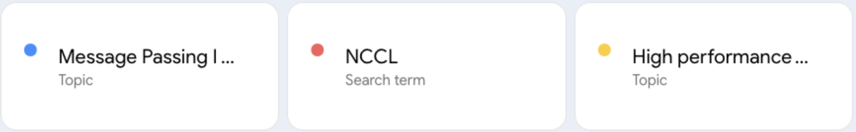


NEW ADDED HPL PERFORMANCE PER LIST



MPI & HPC Over Time


Interest over time

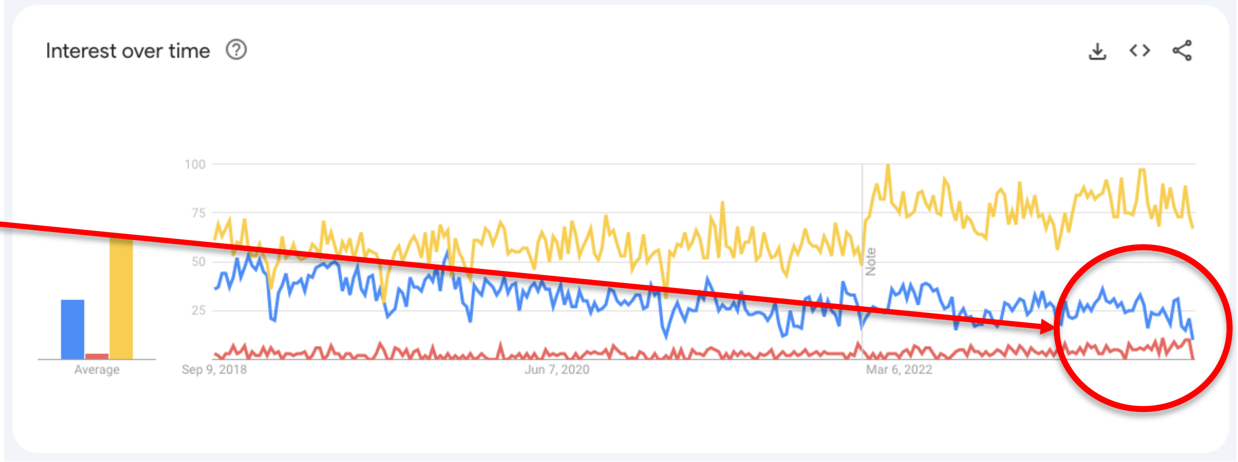

● Message Passing I ...
Topic

● NCCL
Search term

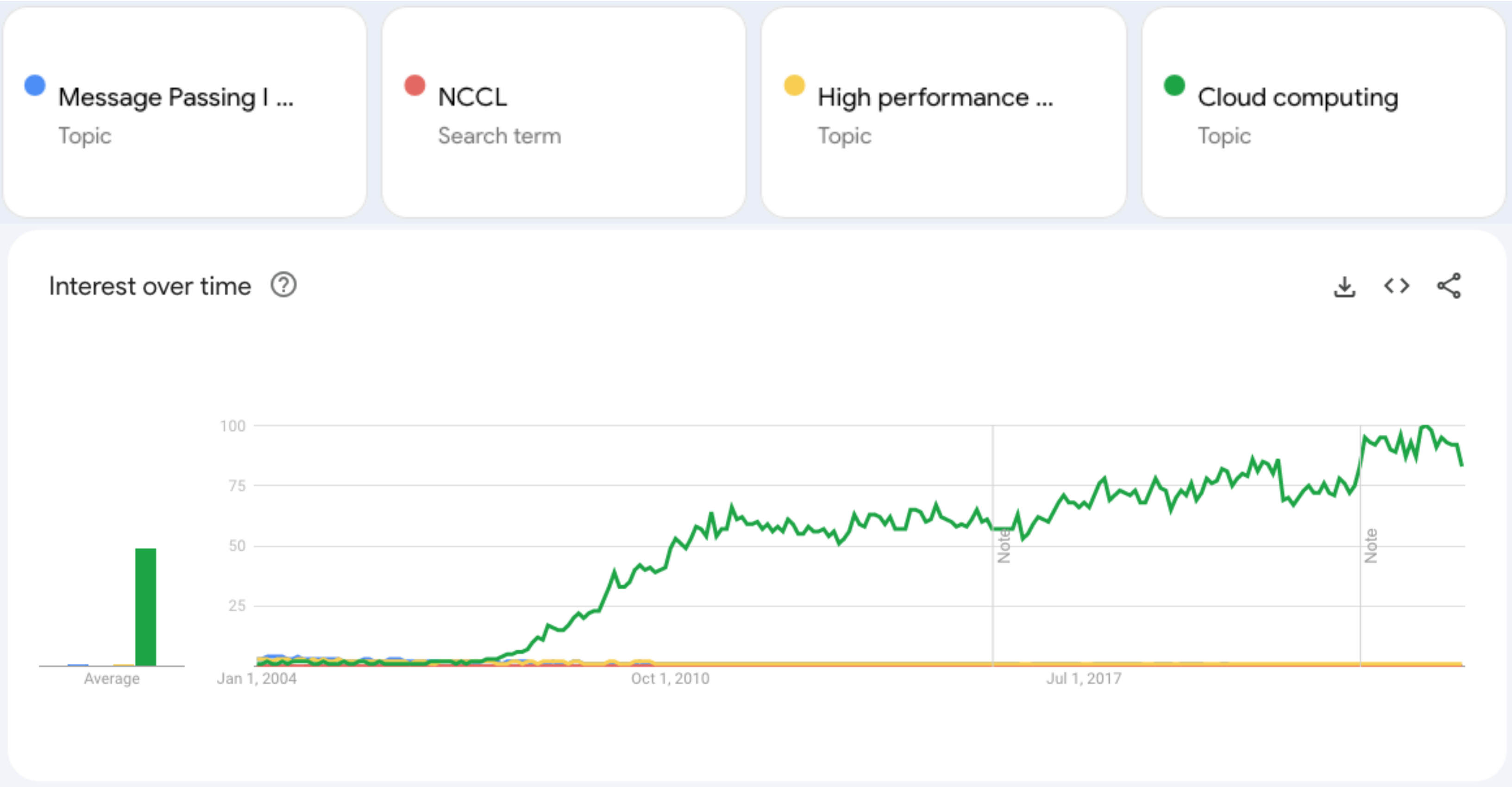
● High performance ...
Topic

Is MPI becoming irrelevant?

(Or everyone just knows about MPI by now?)

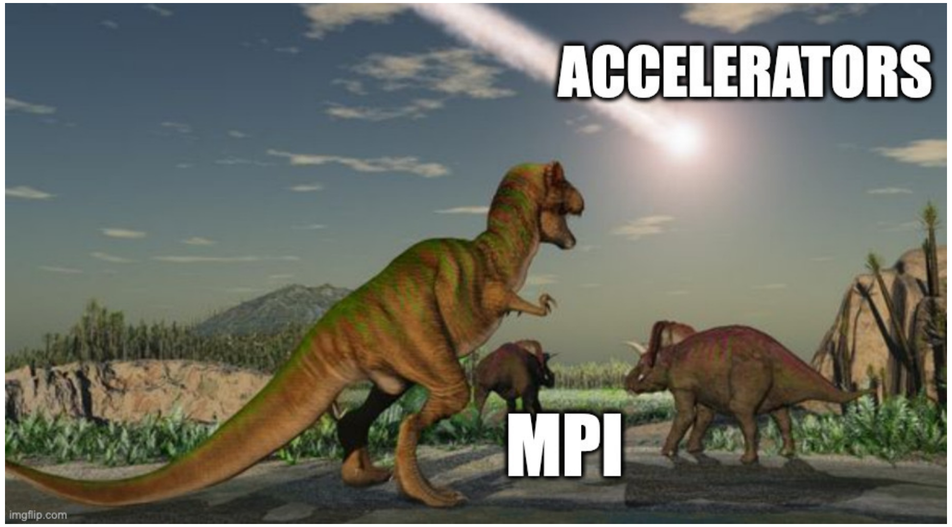

Interest over time

HPC vs Cloud Computing



The Computing Landscape is Changing

- What can MPI learn from commercial approaches?
- What does MPI bring to the table?
- How can MPI stay relevant?
- What can we learn from the past?



MPI: A History of Stability

The MPI standard is

- A **consistent & stable framework**
- Covering **many aspects** of distributed memory programming
- Nurturing a mature **tools environment**
- **Community-driven**
- **Research-driven**
- Mostly funded through research
- Catering to traditional HPC (academia & HPC Centers)

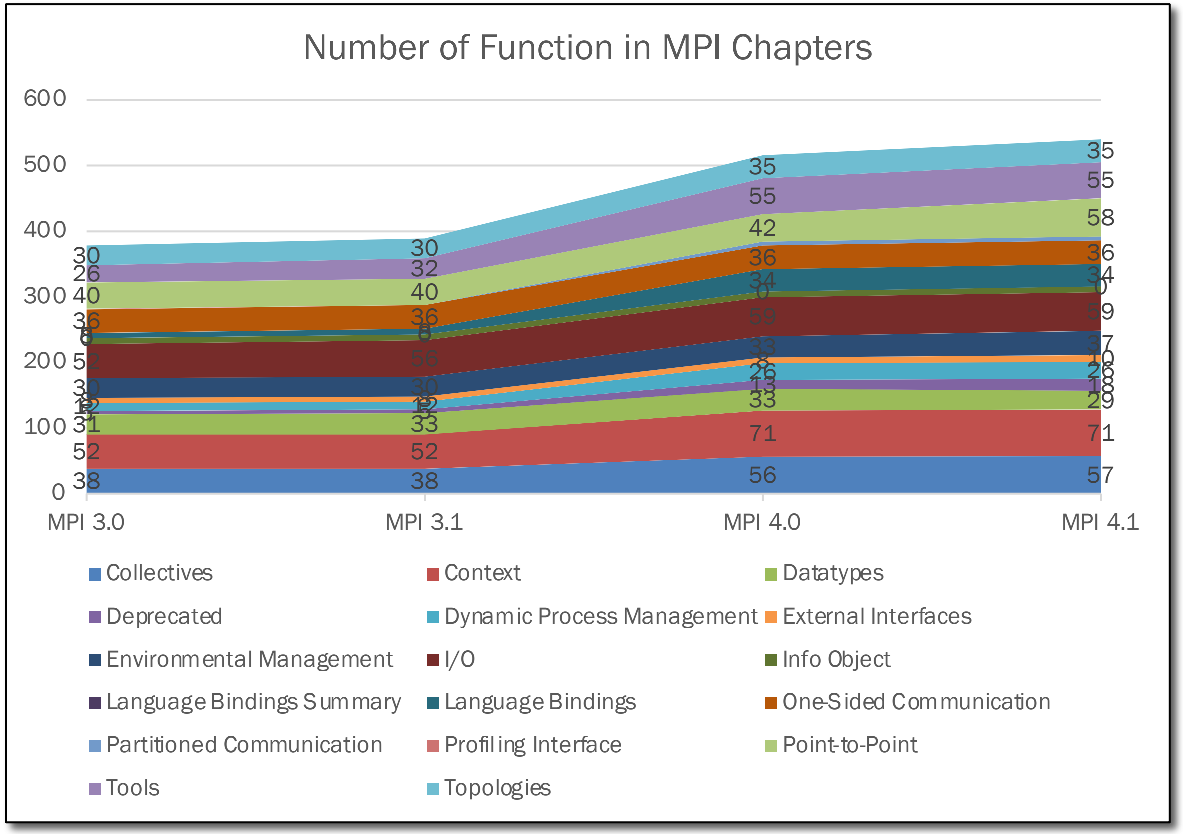
The MPI standard is not

- Moving fast
- Compact
- Easy to extend and adapt
- Removing features easily
- Catering to broader demands

MPI Chapters by the Numbers

- MPI 4.1: 540 functions
  - 4.0 – 4.1: +2.9%
  - 3.1 – 4.0: +32%
  - 4.0 – 4.1: +4.6%
  - Includes deprecated functions (5-18)
  - Does not include big-count & PMPI

NCCL: 28 functions (v2.18)



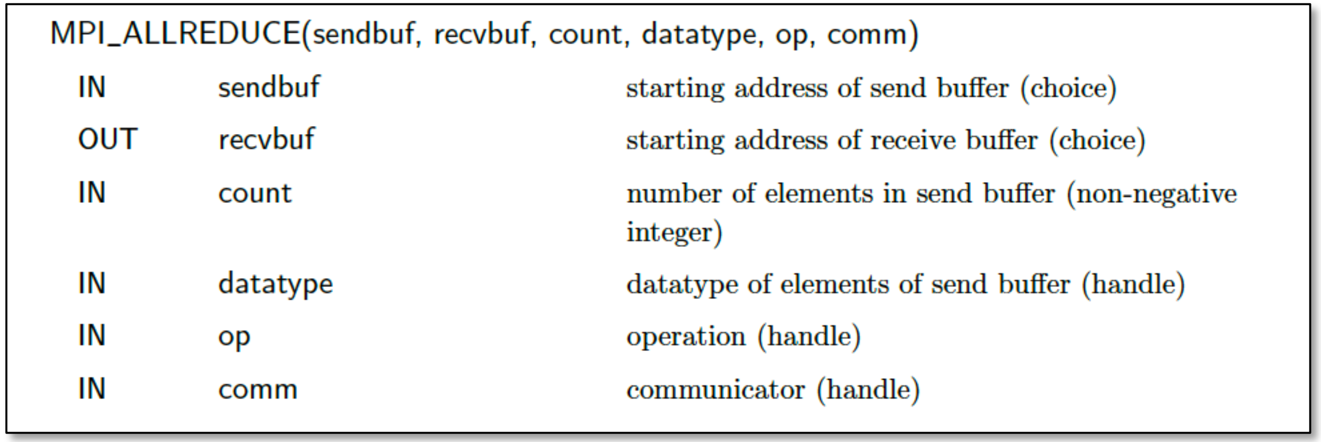
`<4.0: grep ‘funcdef’; >=4.0: grep ‘mpi-binding’`

Comparing Communication Libraries

Feature	MPI 4.1	(NV)SHMEM	NCCL
Communicators, Groups	✅	✅	✅
Custom Reduction Operators	✅	❌	✅
Collective Communication	✅	✅	✅
P2P Communication	✅	❌	✅
Profiling API (call interposition)	✅	✅	✅
One-Sided Communication	✅	✅	❌
Tool Introspection	✅	❌	❌
Custom Datatypes	✅	❌	❌
Multi-Library Support	✅	❌	❌
Failure Mitigation	❌	❌	✅
Stream-aware communication	❌	❌	✅
Device-Side Communication	❌	✅	❌

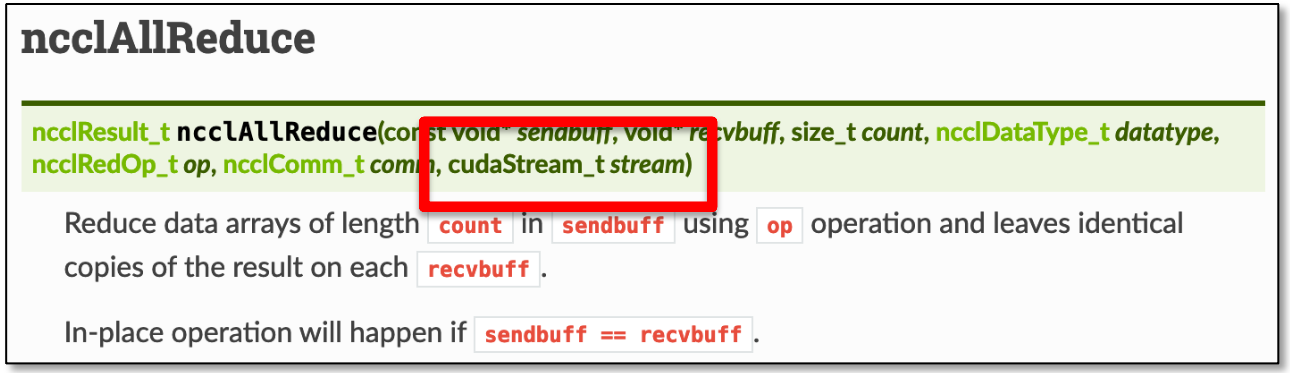
Other Alternatives:
Gloo
MapReduce

Example: Allreduce



MPI_ALLREDUCE(sendbuf, recvbuf, count, datatype, op, comm)

IN	sendbuf	starting address of send buffer (choice)
OUT	recvbuf	starting address of receive buffer (choice)
IN	count	number of elements in send buffer (non-negative integer)
IN	datatype	datatype of elements of send buffer (handle)
IN	op	operation (handle)
IN	comm	communicator (handle)



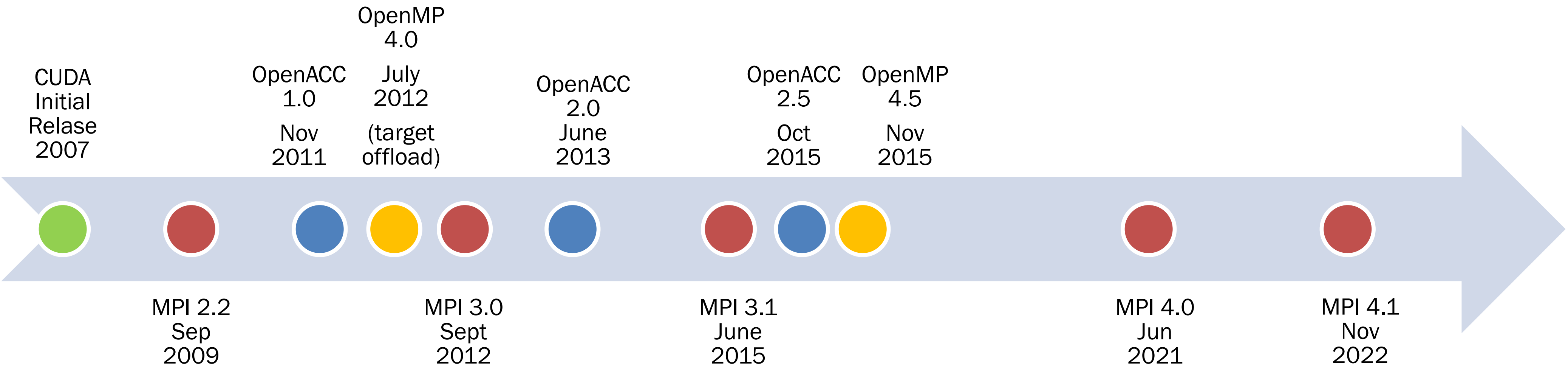
ncclAllReduce

ncclResult_t **ncclAllReduce**(const void* *sendbuff*, void* *recvbuff*, size_t *count*, ncclDataType_t *datatype*, ncclRedOp_t *op*, ncclComm_t *comm*, cudaStream_t *stream*)

Reduce data arrays of length `count` in `sendbuff` using `op` operation and leaves identical copies of the result on each `recvbuff`.

In-place operation will happen if `sendbuff == recvbuff`.

An Analogy? OpenMP and OpenACC



CUDA Initial Relase 2007

MPI 2.2 Sep 2009

OpenACC 1.0 Nov 2011

OpenMP 4.0 July 2012 (target offload)

MPI 3.0 Sept 2012

OpenACC 2.0 June 2013

MPI 3.1 June 2015

OpenACC 2.5 Oct 2015

OpenMP 4.5 Nov 2015

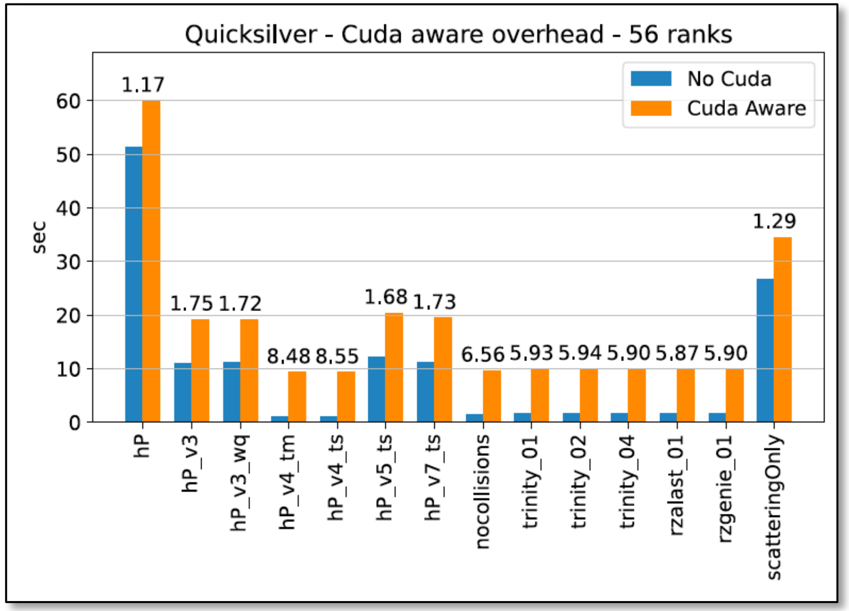
MPI 4.0 Jun 2021

MPI 4.1 Nov 2022

Device Support in MPI 4.1

- **Enabling**/requesting support for certain memory spaces during startup/initialization
- **Asserting** usage of memory spaces in communication
- **Side document** describing memory spaces
- Hybrid & Accelerator WG (Jim Dinan)

Goal: avoid unnecessary initialization and buffer checks

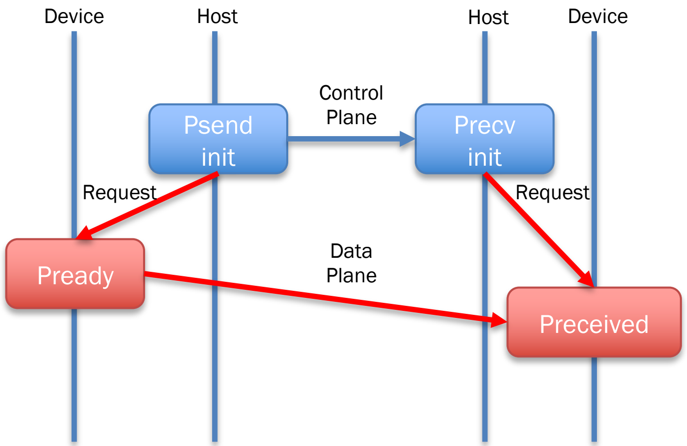


M. Moraru et al:. Benefits of MPI Sessions for GPU MPI applications. *EuroMPI '21 - 28th European MPI Users' Group Meeting*, Sep 2021.

Device Support Beyond MPI 4.1

Device-side triggered communications

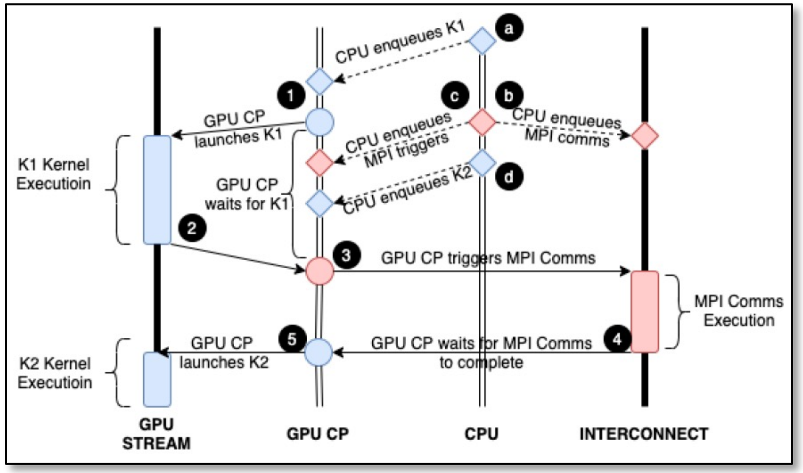
- Partitioned communication to separate control plane (CPU) and data plane (device)
- Missing from 4.1: Request transfer to device & RTS/CTS signaling
- **But:** Partitioned communication is not a panacea
  - Static communication patterns
  - Dynamic pattern need different approaches



Device Support Beyond MPI 4.1

Stream-synchronous communication

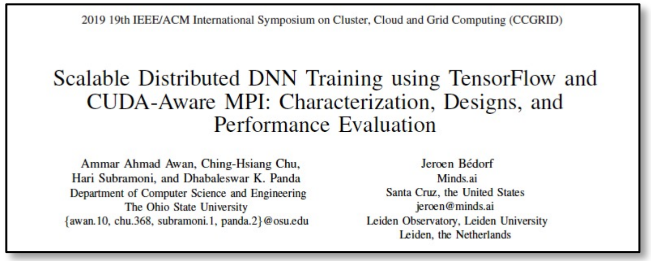
- Make MPI aware of device streams
- Order communication with computation on stream
- Several proposals to consolidate:
  - MPIX_Streams
  - MPIX_Queue
  - Graph Execution Engine



N. Namashivayam et al: Exploring GPU Stream-Aware Message Passing using Triggered Operations. 2022. https://doi.org/10.48550/arXiv.2208.04817

Device Support in Implementations

- Implementations slowly added CUDA for reductions
- MPI can achieve performance similar to NCCL
- Why do only 2 implementations support device offload of reductions?
  - Mostly engineering effort
  - Conflict with proprietary network libraries

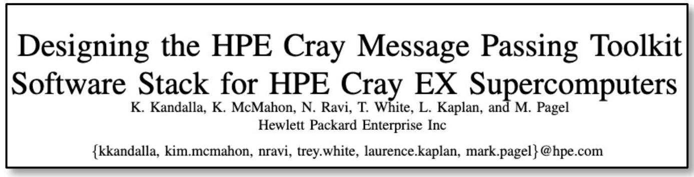


2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)

Scalable Distributed DNN Training using TensorFlow and CUDA-Aware MPI: Characterization, Designs, and Performance Evaluation

Ammar Ahmad Awan, Ching-Hsiang Chu,
Hari Subramoni, and Dhabaleswar K. Panda
Department of Computer Science and Engineering
The Ohio State University
{awan.10, chu.368, subramoni.1, panda.2}@osu.edu

Jeroen Bédorf
Minds.ai
Santa Cruz, the United States
jeroen@minds.ai
Leiden Observatory, Leiden University
Leiden, the Netherlands



Designing the HPE Cray Message Passing Toolkit Software Stack for HPE Cray EX Supercomputers

K. Kandalla, K. McMahon, N. Ravi, T. White, L. Kaplan, and M. Pagel
Hewlett Packard Enterprise Inc
{kkandalla, kim.mcmahon, nravi, trey.white, laurence.kaplan, mark.pagel}@hpe.com

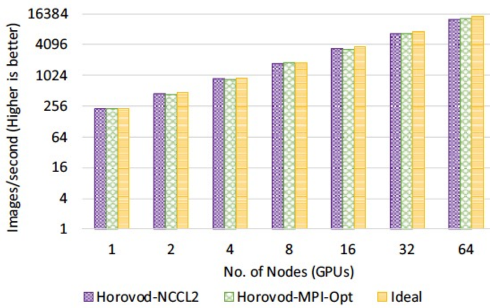
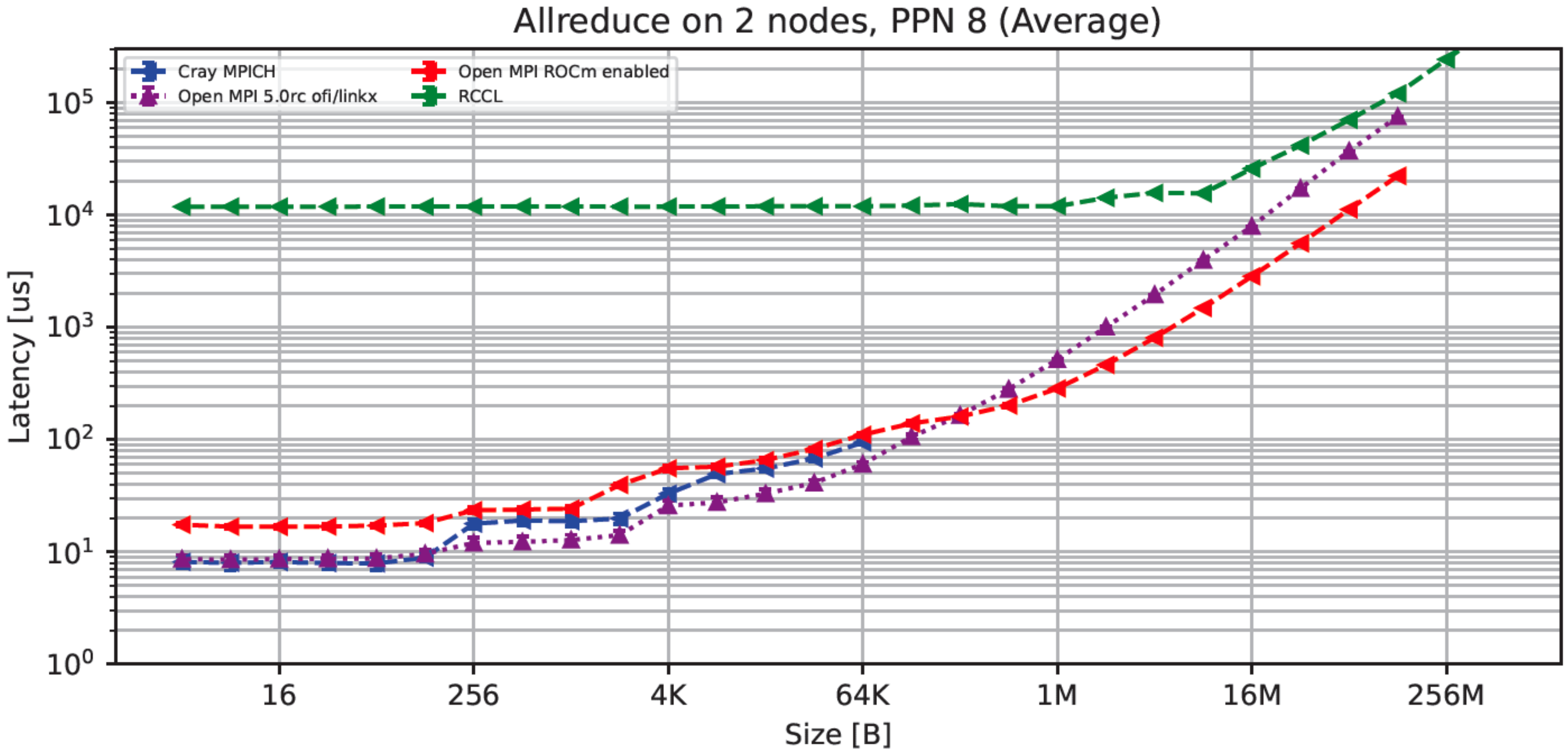


Fig. 9. Performance comparison for ResNet-50: Training performed using two Horovod designs on the Owens Cluster (up to 64 GPUs). 1) NCCL 2.3.4 was used for NCCL experiments. 2) Horovod-MPI-Opt refers to the design that takes advantage of the new Allreduce implementation made available in the MVAPICH2-GDR 2.3rc1 library.

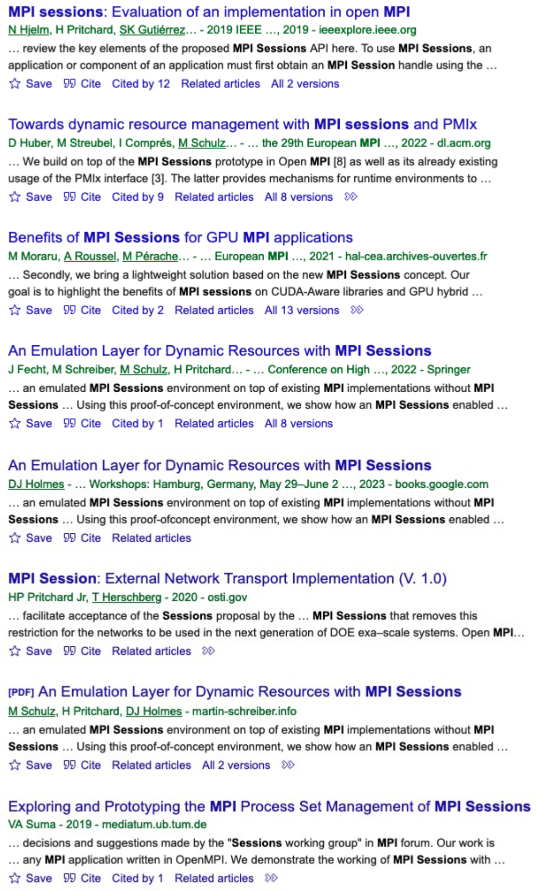
Allreduce on Frontier



Allreduce on Frontier (RCCL, Open MPI (unofficial), Cray MPICH)

Sessions: Make it count

- Sessions become available in implementations
- Many envisioned features have not materialized [1]
  - Resource isolation
  - Fault Tolerance
- Uptake by applications?
  - Too early to say
- Mainly vehicles for malleability research?



MPI sessions: Evaluation of an implementation in open **MPI**
N Hjelm, H Pritchard, SK Gutiérrez… - 2019 IEEE …, 2019 - ieeexplore.ieee.org
… review the key elements of the proposed **MPI Sessions** API here. To use **MPI Sessions**, an application or component of an application must first obtain an **MPI Session** handle using the …

Towards dynamic resource management with **MPI sessions** and PMIx
D Huber, M Streubel, I Comprés, M Schulz… - … the 29th European **MPI** …, 2022 - dl.acm.org
… We build on top of the **MPI Sessions** prototype in Open **MPI** [8] as well as its already existing usage of the PMIx interface [3]. The latter provides mechanisms for runtime environments to …

Benefits of **MPI Sessions** for GPU **MPI** applications
M Moraru, A Roussel, M Pérache… - … European **MPI** …, 2021 - hal-cea.archives-ouvertes.fr
… Secondly, we bring a lightweight solution based on the new **MPI Sessions** concept. Our goal is to highlight the benefits of **MPI sessions** on CUDA-Aware libraries and GPU hybrid …
☆ Save 99 Cite Cited by 2 Related articles All 13 versions »

An Emulation Layer for Dynamic Resources with **MPI Sessions**
J Fecht, M Schreiber, M Schulz, H Pritchard… - … Conference on High …, 2022 - Springer
… an emulated **MPI Sessions** environment on top of existing **MPI** implementations without **MPI Sessions** … Using this proof-of-concept environment, we show how an **MPI Sessions** enabled …

An Emulation Layer for Dynamic Resources with **MPI Sessions**
DJ Holmes - … Workshops: Hamburg, Germany, May 29–June 2 …, 2023 - books.google.com
… an emulated **MPI Sessions** environment on top of existing **MPI** implementations without **MPI Sessions** … Using this proof-ofconcept environment, we show how an **MPI Sessions** enabled …

MPI Session: External Network Transport Implementation (V. 1.0)
HP Pritchard Jr, T Herschberg - 2020 - osti.gov
… facilitate acceptance of the **Sessions** proposal by the … **MPI Sessions** that removes this restriction for the networks to be used in the next generation of DOE exa–scale systems. Open **MPI**…

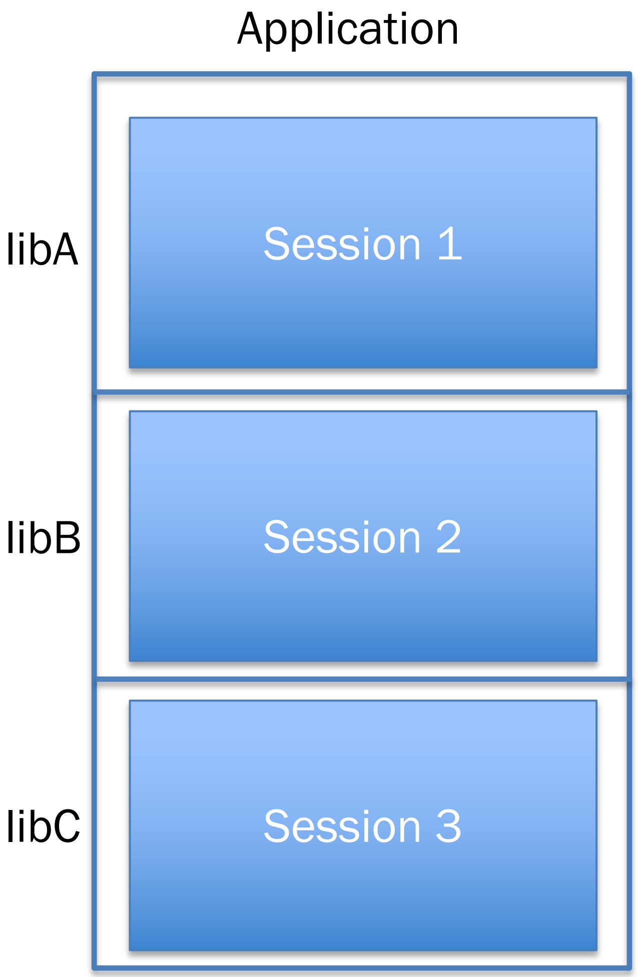
[PDF] An Emulation Layer for Dynamic Resources with **MPI Sessions**
M Schulz, H Pritchard, DJ Holmes - martin-schreiber.info
… an emulated **MPI Sessions** environment on top of existing **MPI** implementations without **MPI Sessions** … Using this proof-of-concept environment, we show how an **MPI Sessions** enabled …

Exploring and Prototyping the **MPI** Process Set Management of **MPI Sessions**
VA Suma - 2019 - mediatum.ub.tum.de
… decisions and suggestions made by the "**Sessions** working group" in **MPI** forum. Our work is … any **MPI** application written in OpenMPI. We demonstrate the working of **MPI Sessions** with …

[1] Holmes, Daniel, et al. "MPI Sessions: Leveraging Runtime Infrastructure to Increase Scalability of Applications at Exascale." *Proceedings of the 23rd European MPI Users' Group Meeting*. 2016.

Sessions Going Forward

- Isolation of Sessions
  - Progress
  - Multi-threading support
  - Resource usage
- Unbound objects
  - Datatypes
  - Attributes
  - Info objects
- Unbound functions?
  - Wait/Test bound through requests, but weak isolation guarantees
- FT-Integration



Application

libA

Session 1

libB

Session 2

libC

Session 3

> Today, MPI’s error handling model is what it has always been; you can assign an error handler to be called when an error occurs in an MPI program, and when that happens you can… well, you can print a nice message before you crash, instead of crashing without the nice message.
>
> [J. Dursi: HPC is dying, and MPI is killing it. 2015]

MPI-4 Error Handling Evolutions

- As part of MPI-4, we introduced changes that makes error handling more ‘localized’

- Initial error handler: set the error handling during mpiexec (to avoid FATAL behavior during MPI Init)
- MPI_ERRORS_ABORT (localize errors to the current comm)
- Errors routed to MPI_COMM_SELF rather than MPI_COMM_WORLD (localize non-comm errors to the local process)

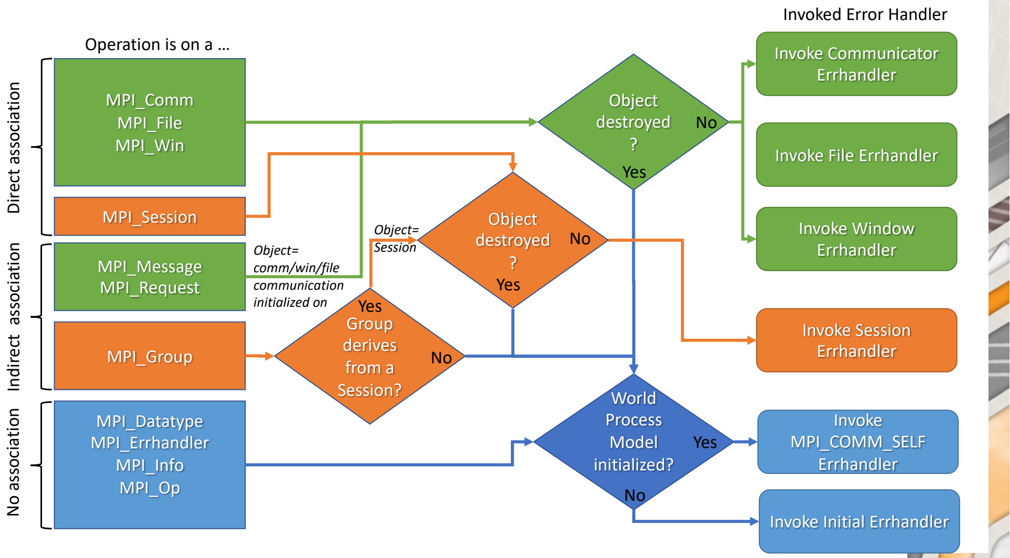
- Overarching goal is that MPI errors would behave more like “Posix” errors
  - Error indicate that the particular operation failed
  - The rest of MPI is not necessarily in a “broken” state
  - Errors should be as local as possible

Courtesy A. Boutellier

Error Handling MPI 4.1 items

- MPI_COMM_CREATE_FROM_GROUP (Issue 511) DONE
  - Error handling changed from 4.0 (Errata)
  - Errors during the operation raised on the Session/Initial error handler
  - Error handler argument is set on the created communicator
- Clarification of error handling fallback (Issue 588) VOTING
  - We found that figuring out where to raise an error (e.g., on a comm, a session, or fallback) was not clear
  - Added flow diagram that clarifies
- MPI_ERR_ERRHANDLER (Issue 525) DONE
  - New error for when an invalid error handler handle is passed to an MPI procedure
- MPI_Delete_error_class/code/string (Issue 283) READING
  - New capability to remove user defined error management handles

Error handling fallback diagram



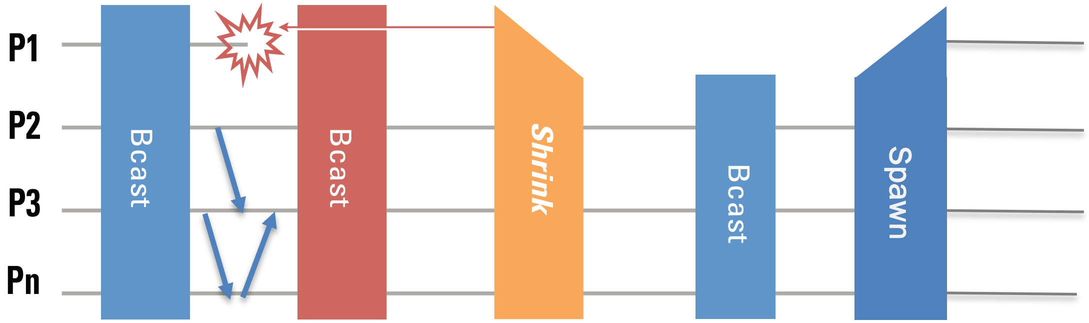
Courtesy A. Boutellier

What's Next? Towards MPI-5

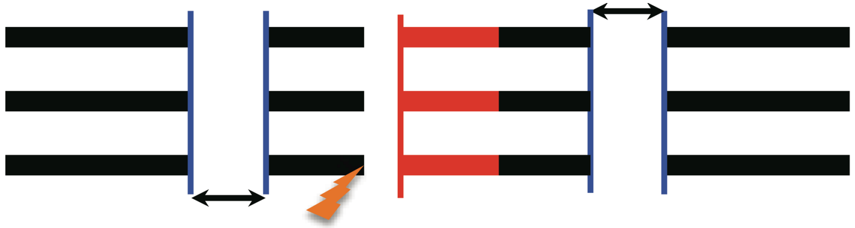
- Current status with "posix-like error handling" gives fallback from MPI errors, crude fault tolerance, *but no MPI fault recovery*

- Working on two main proposals:
  - Fine-grained recovery: "ULFM v2" – Led by Aurelien Bouteiller
  - Coarse-grained recovery: "Reinit" – Led by Ignacio Laguna
  - Proposals designed independently, but designed to be compatible and completement each other

- Implementation Status
  - ULFM v1 and v2 in Open MPI v5.0.x/main
  - ULFM v1 in MPICH
  - Reinit in Open MPI branch



ULFM FT mode:
operations can continue on failure-damaged communicators
SHRINK operation can create new clean communicators without failed processes
Replacement process spawning under user control



Reinit FT mode:
Faults cause the application to return to the MPI_Reinit call
Replacement processes spawned implicitly
All communicators invalidated

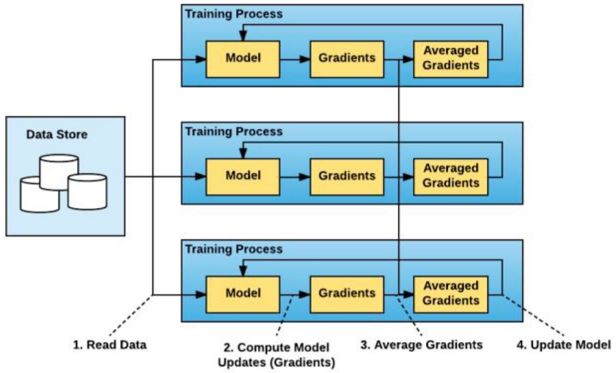
Upcoming new FT features Timetable

- ULFM Slice 1: General Chapter Structure and error reporting
  - MPI_ERR_PROC_FAILED, MPI_Comm_get_failed, MPI_Comm_ack_failed, MPI_Comm_revoke
  - Implicit control for uniformity (same error raised at all ranks in collectives)
  - Implicit control for error range (error raised per-operation/group/universe)
  - VOTED-IN! (Q1/23)
- ULFM Slice 2: MPI_COMM_(I)AGREE
  - New interface removes linkage with "ack_failed" (cleaner
  - Ready for reading ETA: Q2/23?
- ULFM Slice 3: MPI_COMM_(I)SHRINK
  - Communicator centric mode for creating repaired comms
  - Support spawning replacement in combination with MPI2 Dynamics
  - Ready for reading ETA: Q3/23?
- Slice 4: Query interface for FT mode, --with-ft mpiexec argument
  - Query from the program if an FT mode is available at runtime (code must compile, but FT is expected to be runtime-off by default in most impl.)
  - Prior interface with Attribute on MPI_COMM_WORLD undesirable (incompatible with sessions)
  - New interface required, must support enabling/querying multiple modes (if applicable)
  - Design phase ETA: Q4/23
- Slice 5: MPI 2 Dynamics
  - Old text complex, because we wanted to support fully local model (root-only consistency)
  - Revisit: should we move to a "uniform" model, or "uniform by default" model for dynamics? Text would be simpler, examples too. ETA: Q4/23
- Slice 6: Files
  - Old text probably good ETA: /24
- Slice 7: RMA
  - Old text generally sound, but may need some rework to unify with the wording in Slice 1 ETA: /24
  - Should we have "group" error range by default on Windows?
  - Should we have only "group" error range on Windows?
- Slice 5.5: Sessions and Malleability
  - Define fault behavior for MPI_COMM_CREATE_FROM_GROUP (the main session entrypoint that is not a local operation) Discussion started, ETA: Q4/23
  - Define fault behavior for MPI_SESSION_FREE/DISCONNECT (since these are collective) Discussion started, ETA: Q4/23
  - MPI_SESSION_REVOKE?
  - Shrinking psets? Versionned psets? Discussion started but still nebulous
- Reinit
  - Ignacio will show some text soon (maybe next meeting depending on agenda)
  - Concepts can coexist in both standard and implementation
  - Incorporating both models will require some glue text (not hard)

Courtesy A. Boutellier

Case Study1: MPI in Horovod

- Horovod coordinates allreduce of gradients
- NCCL allreduce executed in stream order
- Horovod designed around these constraints
- Serialized communication in Horovod-MPI



Data Store

Training Process

Model

Gradients

Averaged Gradients

Training Process

Model

Gradients

Averaged Gradients

Training Process

Model

Gradients

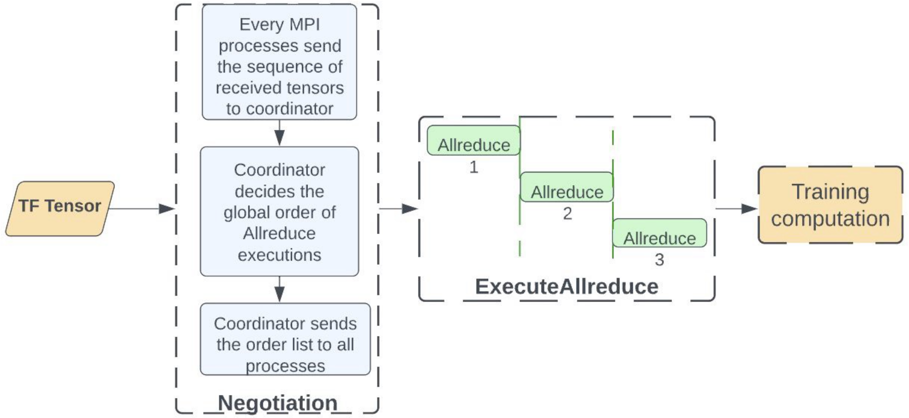
Averaged Gradients

1. Read Data

2. Compute Model Updates (Gradients)

3. Average Gradients

4. Update Model



TF Tensor

Every MPI processes send the sequence of received tensors to coordinator

Coordinator decides the global order of Allreduce executions

Coordinator sends the order list to all processes

Negotiation

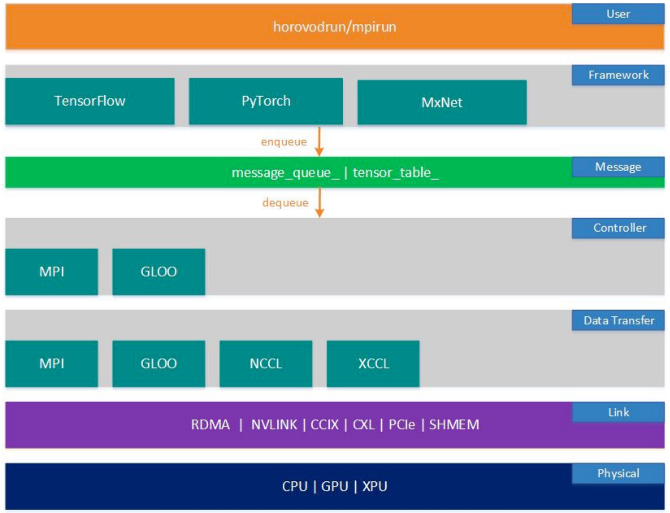
Allreduce 1

Allreduce 2

Allreduce 3

ExecuteAllreduce

Training computation



User

horovodrun/mpirun

Framework

TensorFlow

PyTorch

MxNet

enqueue

Message

message_queue_ | tensor_table_

dequeue

Controller

MPI

GLOO

Data Transfer

MPI

GLOO

NCCL

XCCL

Link

RDMA | NVLINK | CCIX | CXL | PCIe | SHMEM

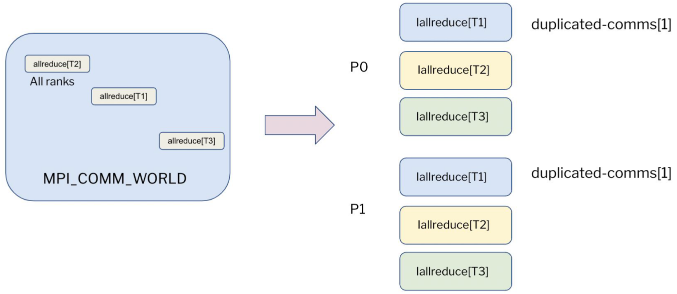
Physical

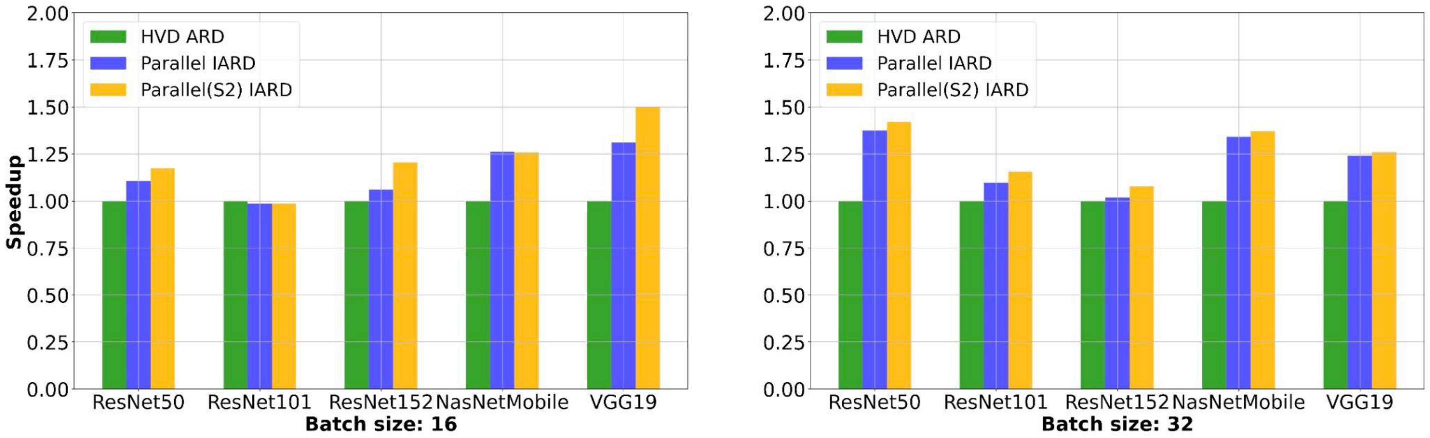
CPU | GPU | XPU

J. Li PhD thesis (2023)

Concurrent Allreduce in Horovod

- Communicators: concurrent collectives
  - Not possible with NCCL
- Avoid negotiation phase in Horovod
- Better utilize network bandwidth
- Improve training throughput up to 50%
  - Over default Horovod-MPI

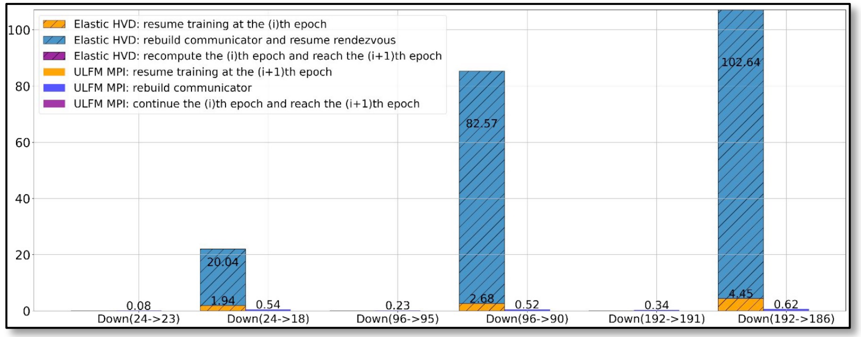


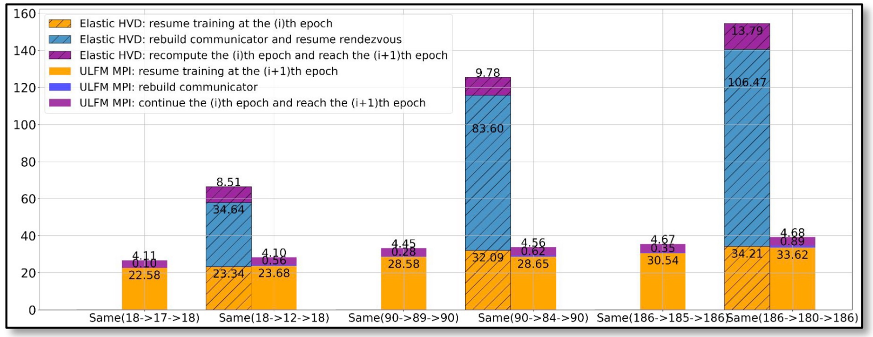


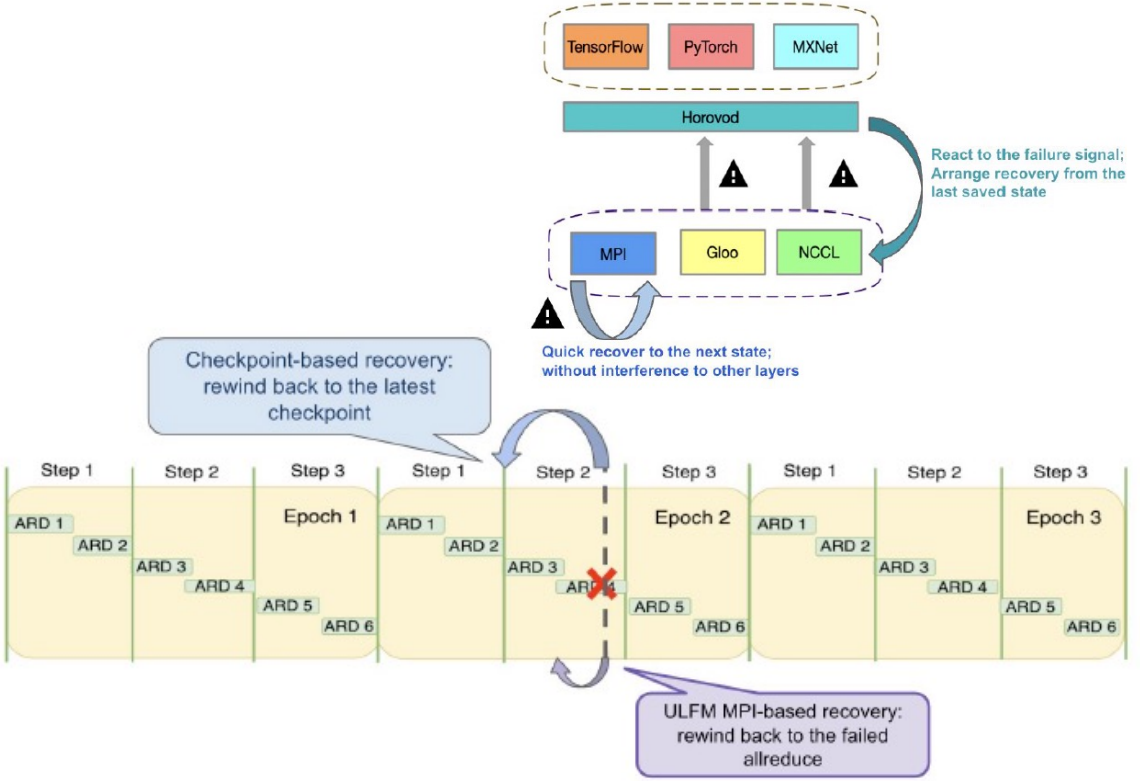
J. Li PhD thesis (2023)

Case Study 2: ULFM & Horovod

- Elastic Horovod: fault mitigation through checkpointing
- ULFM: shrinking & growing of communicators







TensorFlow
PyTorch
MXNet
Horovod
React to the failure signal; Arrange recovery from the last saved state
MPI
Gloo
NCCL
Quick recover to the next state; without interference to other layers
Checkpoint-based recovery: rewind back to the latest checkpoint
Step 1
Step 2
Step 3
Step 1
Step 2
Step 3
Step 1
Step 2
Step 3
Epoch 1
Epoch 2
Epoch 3
ARD 1
ARD 2
ARD 3
ARD 4
ARD 5
ARD 6
ARD 1
ARD 2
ARD 3
ARD 4
ARD 5
ARD 6
ARD 1
ARD 2
ARD 3
ARD 4
ARD 5
ARD 6
ULFM MPI-based recovery: rewind back to the failed allreduce

J. Li PhD thesis (2023)

What MPI brings to the table:

Flexible communication patterns
Fine-grained fault tolerance

Case Study 3: MPI vs LCI in PaRSEC

- PaRSEC emulates AMs using Send/Recv
- MPI: Single thread injection & extraction
  - Request management
  - Multi-threading concerns
  - Opaque progress semantics
- LCI backend: explicit progress threads
  - Improved extraction rate
  - Reduced starvation

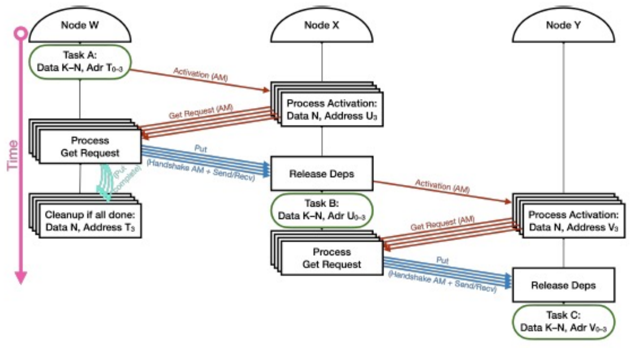
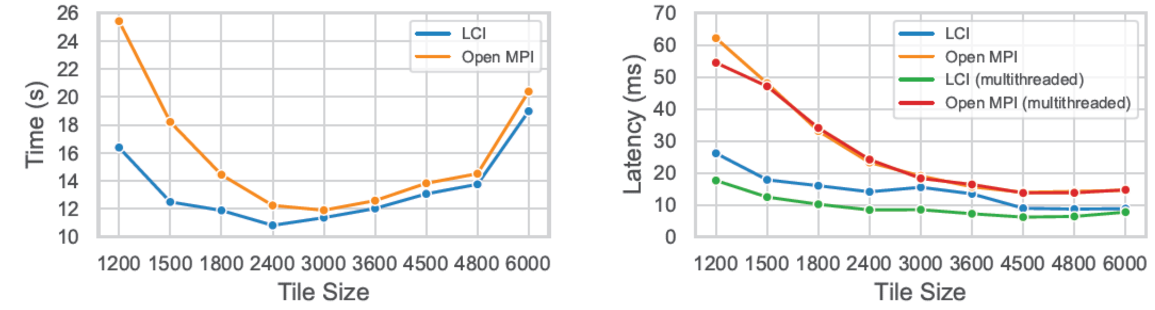

Node W
Node X
Node Y
Time
Task A: Data K–N, Adr T0-3
Activation (AM)
Process Activation: Data N, Address U3
Get Request (AM)
Process Get Request
Put (Handshake AM + Send/Recv)
(Put complete)
Release Deps
Cleanup if all done: Data N, Address T3
Activation (AM)
Task B: Data K–N, Adr U0-3
Process Activation: Data N, Address V3
Get Request (AM)
Process Get Request
Put (Handshake AM + Send/Recv)
Release Deps
Task C: Data K–N, Adr V0-3

Figure 1: PaRSEC communication example. Task *A* runs on node *W* with descendant tasks *B* and *C* on nodes *X* and *Y*, respectively. There are four dataflows that must be propagated as part of the broadcast.



(a) Time-to-solution.

(b) End-to-end communication latency.

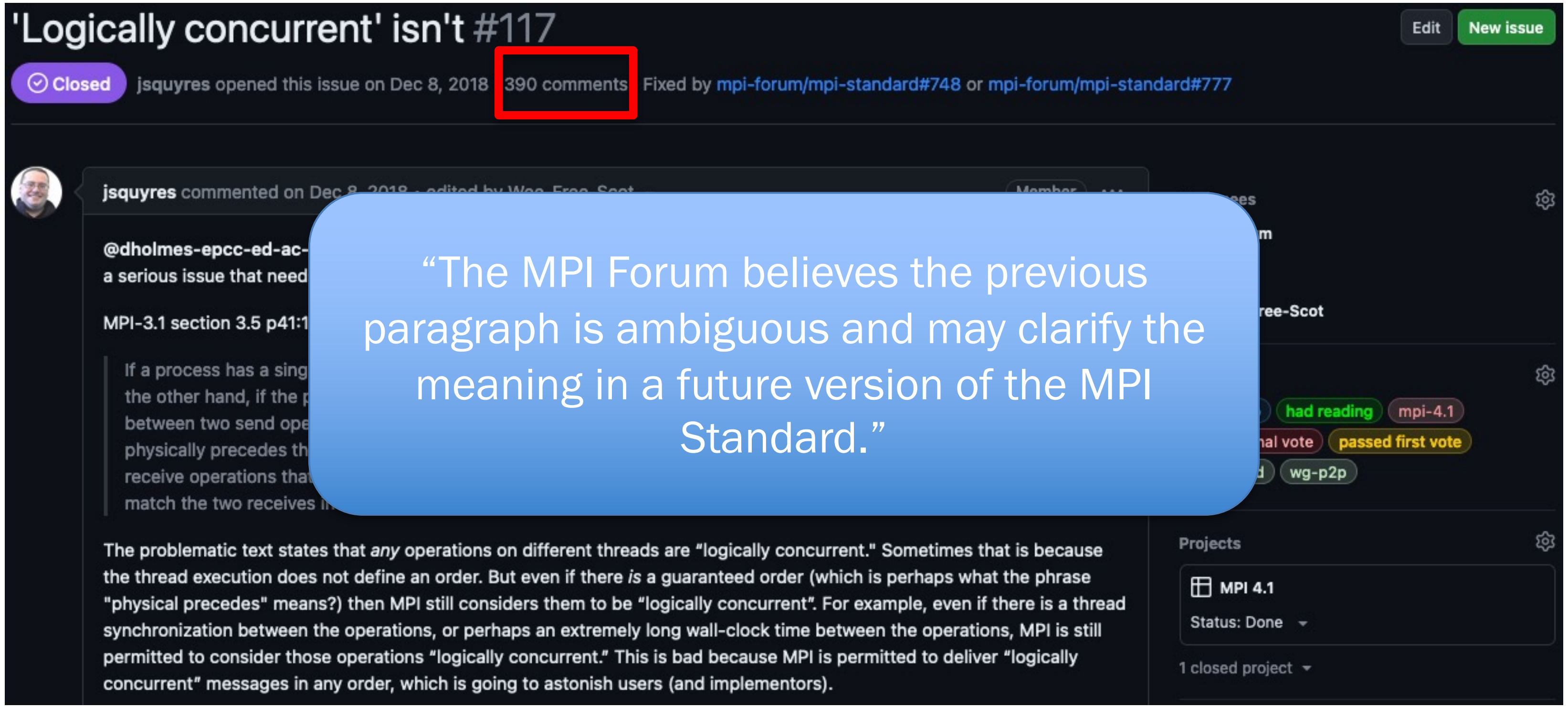
Figure 4: TLR Cholesky, N = 360,000, 16 nodes, scaling tile size from 6000 × 6000 to 1200 × 1200. Latency is measured from send of the ACTIVATE message to arrival of data for individual flows. "MT" indicates that communication multi-threading for ACTIVATE messages is enabled.

Omri Mor, George Bosilca, and Marc Snir. "Improving the Scaling of an Asynchronous Many-Task Runtime with a Lightweight Communication Engine." (2023).

Progress & Threads in MPI

- Definition of progress in MPI 4.1 first step
- **Cooperative strong progress**
  - No application interference
  - But application cooperation
- Revisit previous efforts (MPI teams)
- Ties in with thread-local resources
  - MPIX_Stream, virtual endpoints
  - Improved injection & extraction

Why MPI Progress is slow



'Logically concurrent' isn't #117

jsquyres opened this issue on Dec 8, 2018 · 390 comments · Fixed by mpi-forum/mpi-standard#748 or mpi-forum/mpi-standard#777

jsquyres commented on Dec 8, 2018 · edited by Wee-Free-Scot

@dholmes-epcc-ed-ac-
a serious issue that need

MPI-3.1 section 3.5 p41:1

> If a process has a sing
> the other hand, if the p
> between two send ope
> physically precedes th
> receive operations tha
> match the two receives i

The problematic text states that *any* operations on different threads are “logically concurrent." Sometimes that is because the thread execution does not define an order. But even if there *is* a guaranteed order (which is perhaps what the phrase "physical precedes" means?) then MPI still considers them to be “logically concurrent”. For example, even if there is a thread synchronization between the operations, or perhaps an extremely long wall-clock time between the operations, MPI is still permitted to consider those operations “logically concurrent.” This is bad because MPI is permitted to deliver “logically concurrent” messages in any order, which is going to astonish users (and implementors).

“The MPI Forum believes the previous paragraph is ambiguous and may clarify the meaning in a future version of the MPI Standard.”

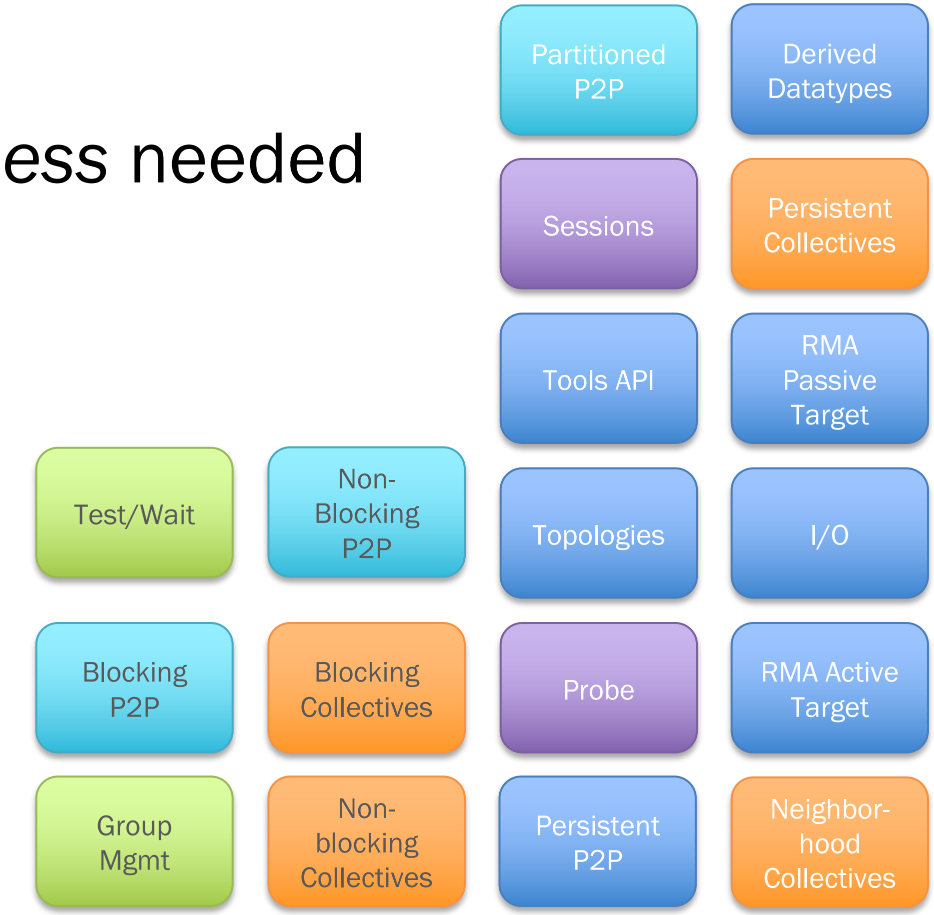
Note: I am not singling out any particular person or institution here.

Where to go from here?

- MPI thrived on stability
- But: past additions were incomplete
- Process for *Extensibility* and *responsiveness* needed

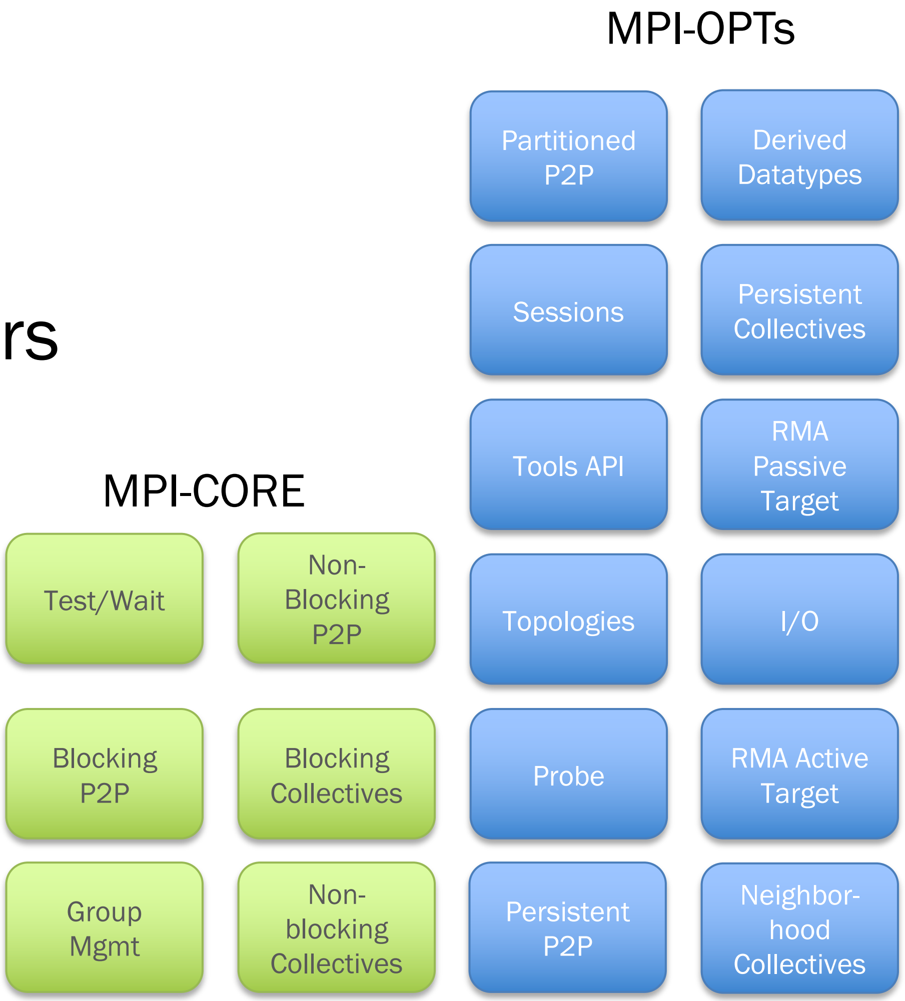
Proposals

- Modularization
- Extensions
- Modernization



Modularizing MPI

- 2 full implementations of MPI
- Big lift for new implementations
  - OMPI: 350kloc for MPI (w/o comments)
- Some features used by only few (no?) users
- Restructuring of document:
  - Main document: core functions
  - Annexes: optional functionality
- Path for removal of features dropped by implementations



MPI-CORE

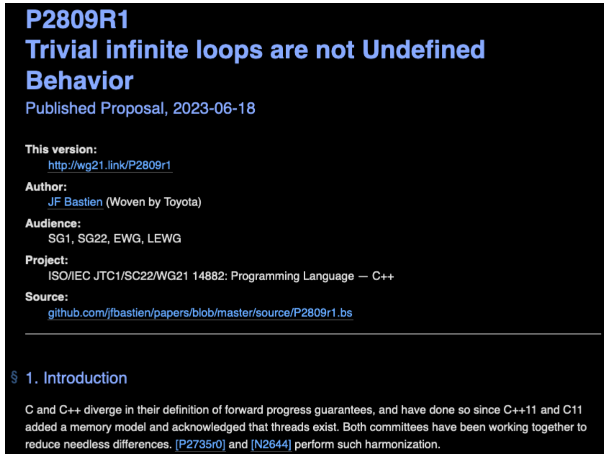
- Test/Wait
- Non-Blocking P2P
- Blocking P2P
- Blocking Collectives
- Group Mgmt
- Non-blocking Collectives

MPI-OPTs

- Partitioned P2P
- Derived Datatypes
- Sessions
- Persistent Collectives
- Tools API
- RMA Passive Target
- Topologies
- I/O
- Probe
- RMA Active Target
- Persistent P2P
- Neighbor-hood Collectives

Process For Extensions

- Formalize procedures and requirements
  1. Official extension namespace
  2. Extension publication (MPI Forum)
  3. Full implementation (MPI Advance?)
  4. Demonstrated use
  5. Upstreaming
- Learn from other communities
  - C & C++
  - OpenMP



P2809R1
Trivial infinite loops are not Undefined Behavior
Published Proposal, 2023-06-18

This version:
http://wg21.link/P2809r1

Author:
JF Bastien (Woven by Toyota)

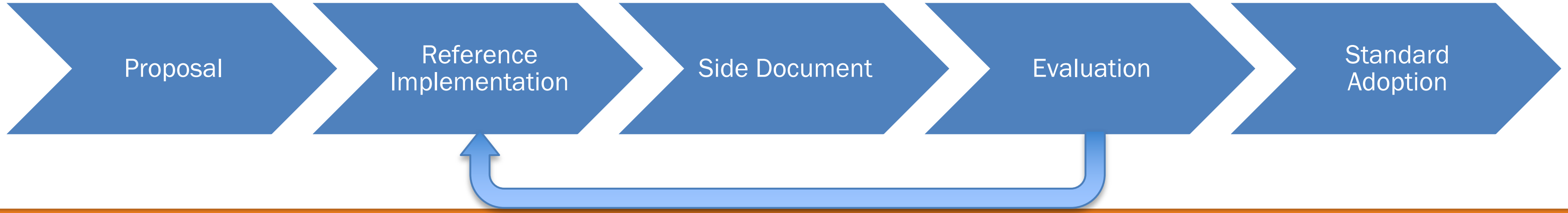
Audience:
SG1, SG22, EWG, LEWG

Project:
ISO/IEC JTC1/SC22/WG21 14882: Programming Language — C++

Source:
github.com/jfbastien/papers/blob/master/source/P2809r1.bs

§ 1. Introduction

C and C++ diverge in their definition of forward progress guarantees, and have done so since C++11 and C11 added a memory model and acknowledged that threads exist. Both committees have been working together to reduce needless differences. [P2735r0] and [N2644] perform such harmonization.



Proposal → Reference Implementation → Side Document → Evaluation → Standard Adoption

Process For Modernization (I)

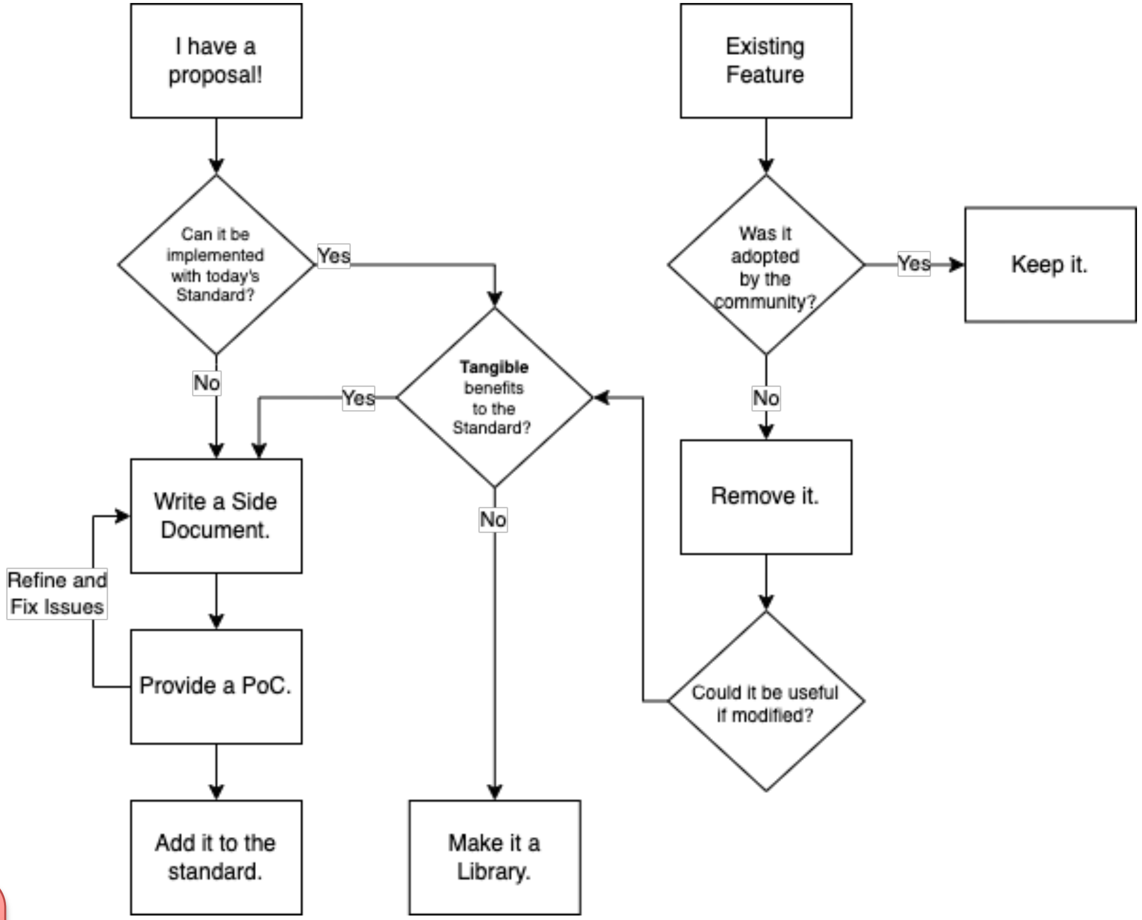
- Standard includes 2 languages that are rooted in traditional HPC: Fortran and C
- Make MPI language-independent
- Language bindings as Side Documents

https://degenerateconic.com/tag/iso.html

Process For Modernization (II)

- MPI on a diet

Cutting off old braids.



I have a proposal!

Can it be implemented with today's Standard?

Yes

No

Tangible benefits to the Standard?

Yes

No

Write a Side Document.

Refine and Fix Issues

Provide a PoC.

Add it to the standard.

Make it a Library.

Existing Feature

Was it adopted by the community?

Yes

Keep it.

No

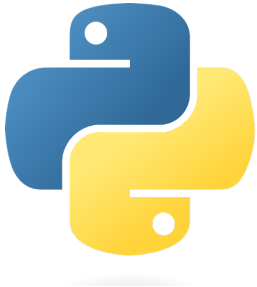
Remove it.

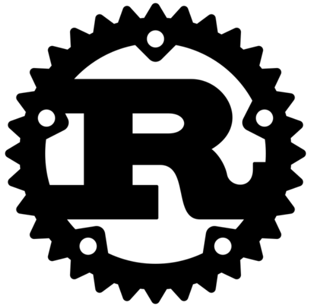
Could it be useful if modified?

How do we know which features were adopted?

Supporting Modern Programming Approaches

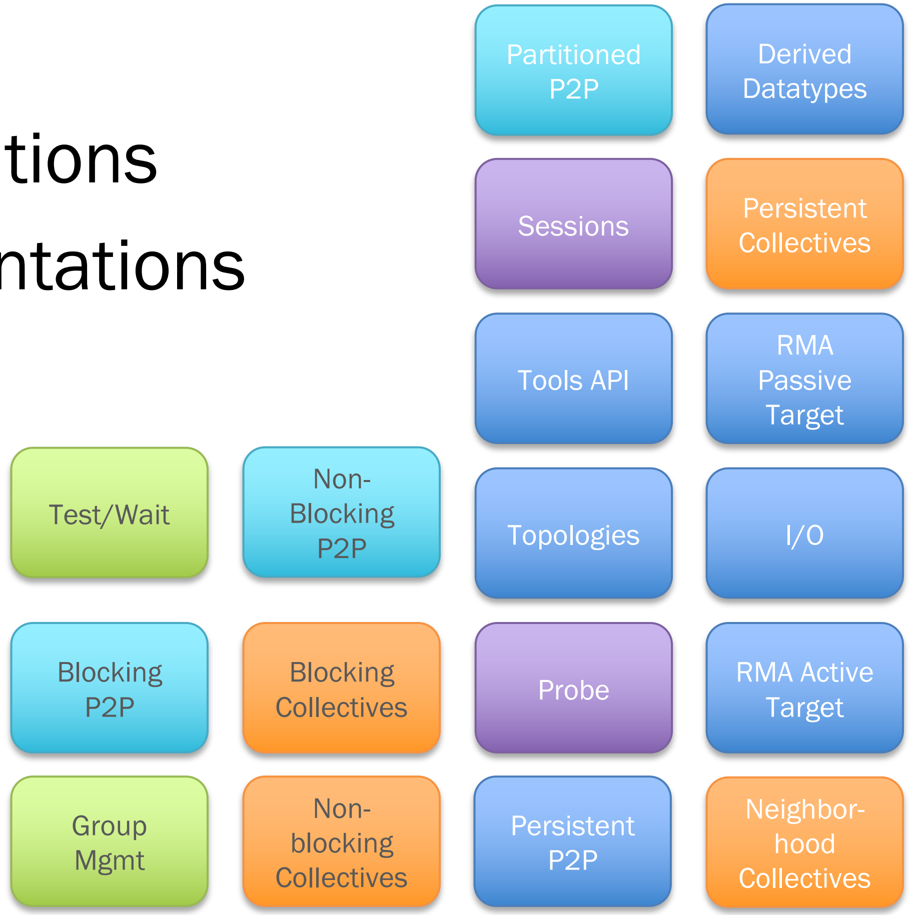
- ABI efforts important step for distribution portability
- Generalized datatypes:
  - Iterables / non-contiguous containers
  - Generators
- Futures: MPI Continuations
  - Side document for 4.1
  - 2 PoC implementations
  - Demonstrated use in applications
  - Proposal finalization





What does MPI bring to the table?

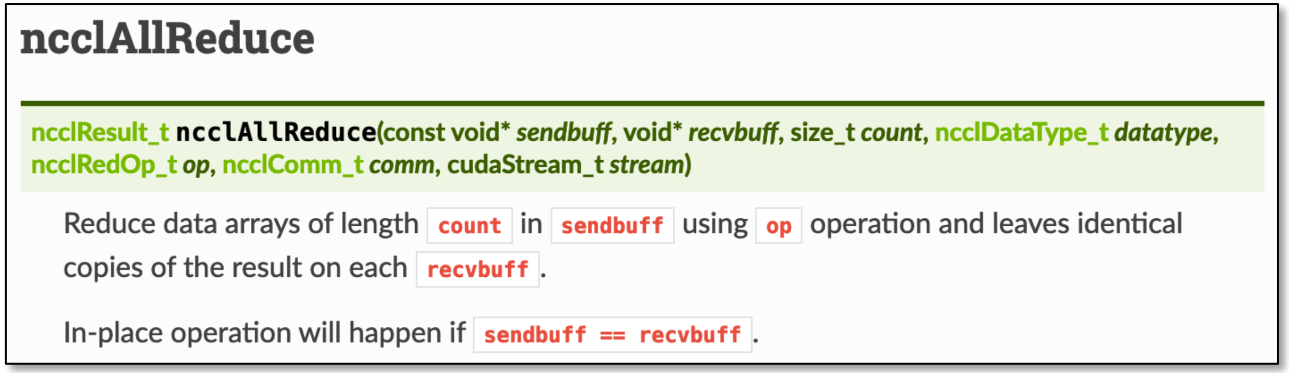
- Wide coverage of operations
- Communication contexts
- Blocking, non-blocking & persistent operations
- A stable API with (mostly) stable implementations
- Decades of experience in HPC



		Partitioned P2P	Derived Datatypes
		Sessions	Persistent Collectives
		Tools API	RMA Passive Target
Test/Wait	Non-Blocking P2P	Topologies	I/O
Blocking P2P	Blocking Collectives	Probe	RMA Active Target
Group Mgmt	Non-blocking Collectives	Persistent P2P	Neighbor-hood Collectives

What to learn from the Competition?

- Adaptation to new paradigms requires (only slight) adjustments
- Accelerators are here to stay
- MPI community must adapt to this reality
- Staging grounds for new features needed



ncclAllReduce

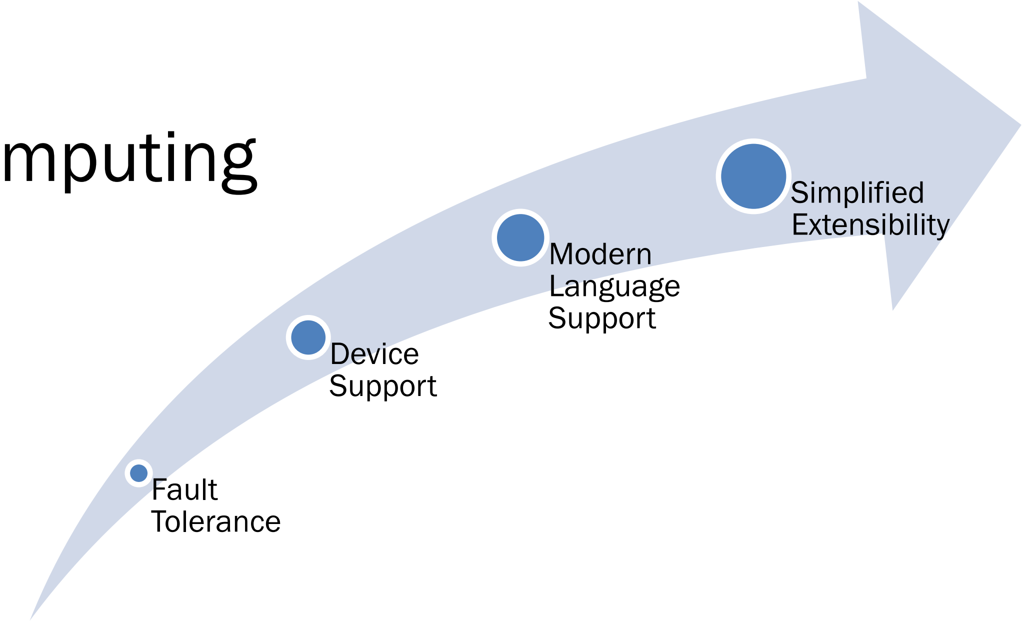
ncclResult_t **ncclAllReduce**(const void* *sendbuff*, void* *recvbuff*, size_t *count*, ncclDataType_t *datatype*, ncclRedOp_t *op*, ncclComm_t *comm*, cudaStream_t *stream*)

Reduce data arrays of length `count` in `sendbuff` using `op` operation and leaves identical copies of the result on each `recvbuff`.

In-place operation will happen if `sendbuff == recvbuff`.

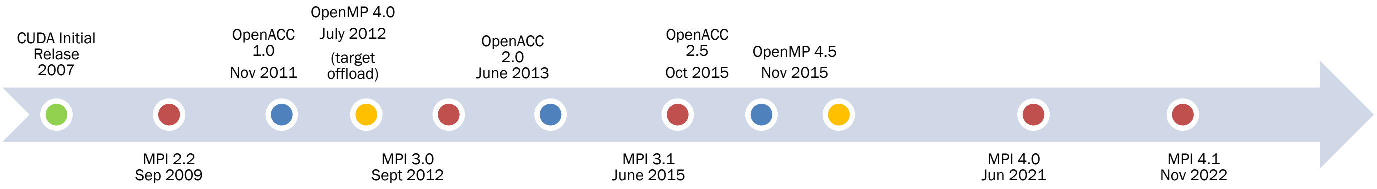
How can MPI stay relevant?

- Fault Tolerance (srsly)
- Device integration:
  - Low-overhead support
  - Device & stream integration
- Support for modern languages
- Timely adaptation to a changing computing eco-systems through extensions
- Focus on current topics



What can be learned from past mistakes?

- Immature proposals should be delayed
  - Without delaying standard releases
- **Regular releases** provide timely updates to users
- Official path for **experimental extensions**
- Stable implementation should be a requirement
- Deliberately slow down expansion of the main standard


CUDA Initial Relase 2007

MPI 2.2 Sep 2009

OpenACC 1.0 Nov 2011

OpenMP 4.0 July 2012 (target offload)

MPI 3.0 Sept 2012

OpenACC 2.0 June 2013

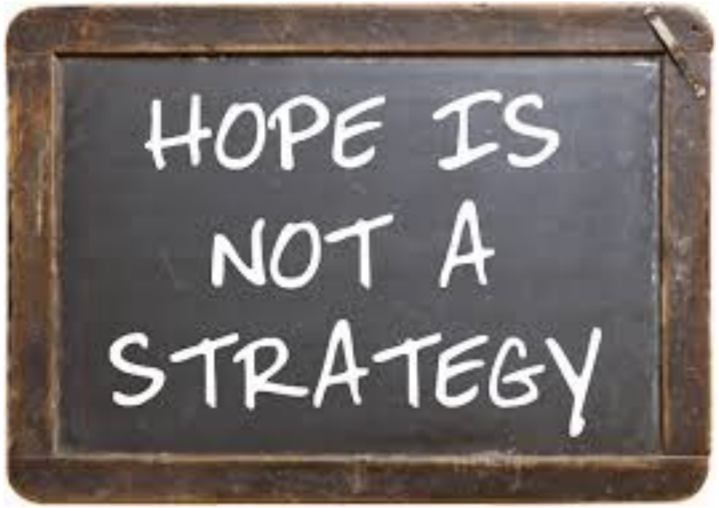
MPI 3.1 June 2015

OpenACC 2.5 Oct 2015

OpenMP 4.5 Nov 2015

MPI 4.0 Jun 2021

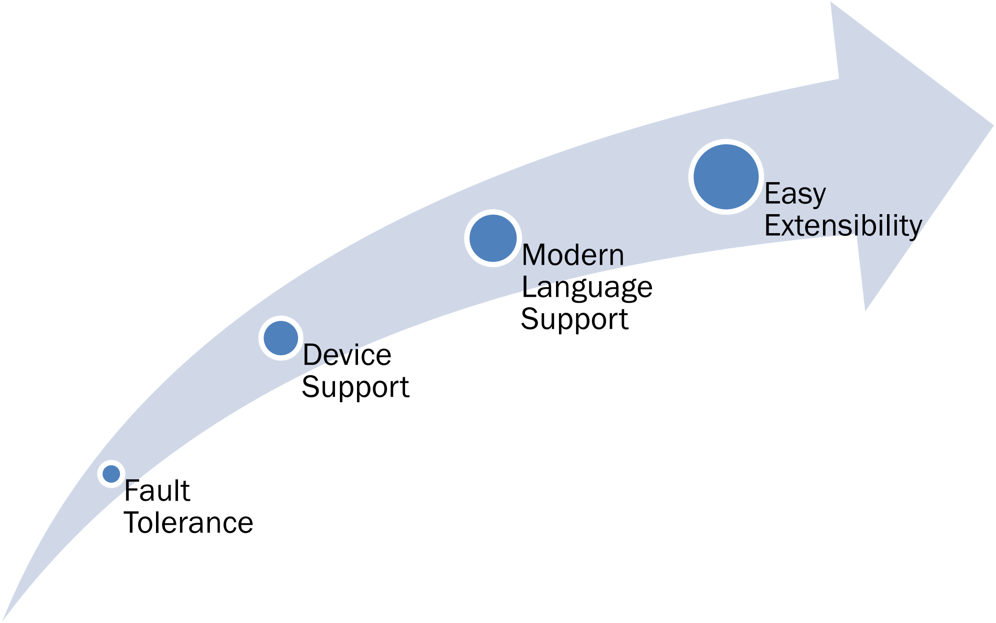
MPI 4.1 Nov 2022



Conclusions

- Manage split between stable framework and adaptable API
- Rethink what communities we focus on
  - Traditional HPC / C & Fortran
  - Modern Languages & compute platforms
- Incorporate advances made in the commercial space
- Deliver solutions for users, not for the MPI standard

Thank You!



Discussion