Experiences with AI-Driven Development in HPC Software

Jeff Hammond
Distinguished Engineer
GPU Communication Software

MVAPICH User Group
August 19, 2026

Motivation

What's the best way for Jeff to evaluate AI development tools?

A. An iPhone app to detect skin cancer

B. A match-matching website for reindeer

C. A Linux driver in Rust

D. Quantum chemistry and MPI stuff

Tools Used Now

- **Claude Code (CLI)**
- **OpenAI Codex (CLI)**
- Cursor IDE and Agent (used for the quantum chemistry project)
- Perplexity (replacement for Google search)
- Claude Co-work (use like Perplexity, or complex local tasks)

I have a Docker container for all my development so I can run in YOLO mode.

VibeMPI

It’s an MPI implementation that sounds like it’s vibe-coded, but isn’t…

VibeMPI Motivation

- Can agentic development be an effective tool for communication software **in general**?
- MPI has a good specification, although it was not intended to be machine-readable (WIP) or machine-comprehensible.
  - An unofficial markdown version of the MPI standard is WIP. I vibe-coded an implementation but nobody has had time to review it for accuracy.
- MPI is widely used, has multiple implementations already, and hundreds of tests.
  - Testing, not the standard, is where the magic happens.

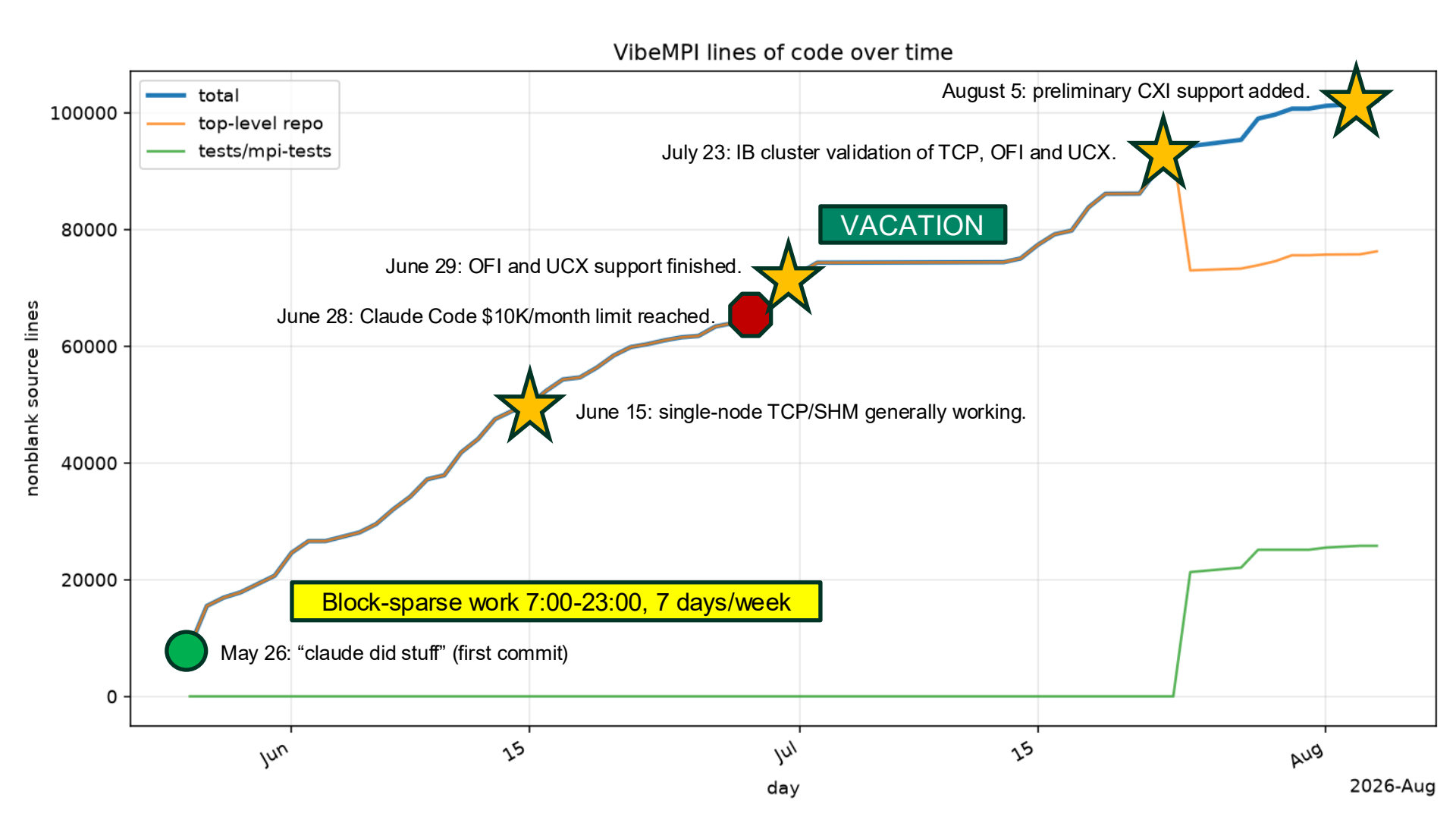
VibeMPI Method

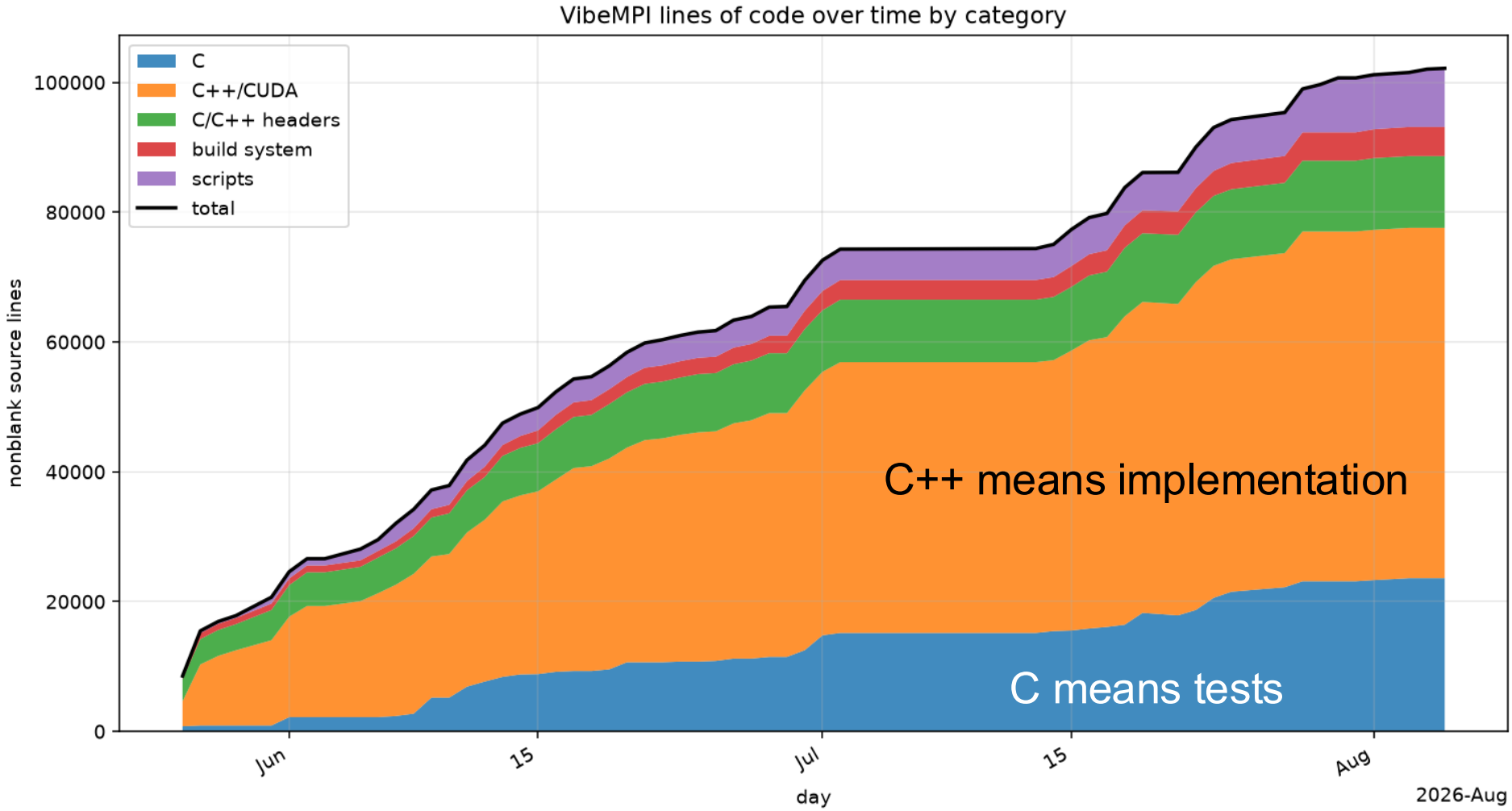
- Collect everything about MPI into a repo to form a RAG database.
  - Scrape all the user lists.
  - Download all the papers.
  - Clone all the implementations (2).
  - Tell the agent to bias towards sources by Gropp, Lusk, Squyres, Panda, Balaji, etc.
- I started writing a book about MPI first. It's interesting but low priority now.
- VibeMPI started simply as a validation test of agentic capability for communication software.

"What I cannot create, I do not understand."
- Richard Feynman

VibeMPI Design

- **Little to no actual vibe-coding.**
  Apply everything the MPICH team has taught me over the years, etc.
  Numerous intensive design phases with N-way experiments on message queues etc.
- No legacy debt. Use modern C++ for the implementation. Assume the latest compilers.
  C++20: concepts/requires, span, filesystem, optional, atomic_ref, stacktrace.
- Design for MPI-5 ABI, MPI_THREAD_MULTIPLE, large-count, fast+correct RMA, etc.
- Use CMake, even though I hate it, because configure+build+test is on the critical path of agentic development and CMake+Ninja moves much faster than Autotools+GNU Make.
- Initial target was SHM and TCP only. OFI and UCX were an afterthought and required a major redesign of the internals to accommodate native tag matching, etc.


VibeMPI lines of code over time
total
top-level repo
tests/mpi-tests
nonblank source lines
100000
80000
60000
40000
20000
0
August 5: preliminary CXI support added.
July 23: IB cluster validation of TCP, OFI and UCX.
VACATION
June 29: OFI and UCX support finished.
June 28: Claude Code $10K/month limit reached.
June 15: single-node TCP/SHM generally working.
Block-sparse work 7:00-23:00, 7 days/week
May 26: “claude did stuff” (first commit)
Jun
15
Jul
15
Aug
day
2026-Aug


VibeMPI lines of code over time by category
C
C++/CUDA
C/C++ headers
build system
scripts
total
nonblank source lines
100000
80000
60000
40000
20000
0
C++ means implementation
C means tests
Jun
15
Jul
15
Aug
day
2026-Aug

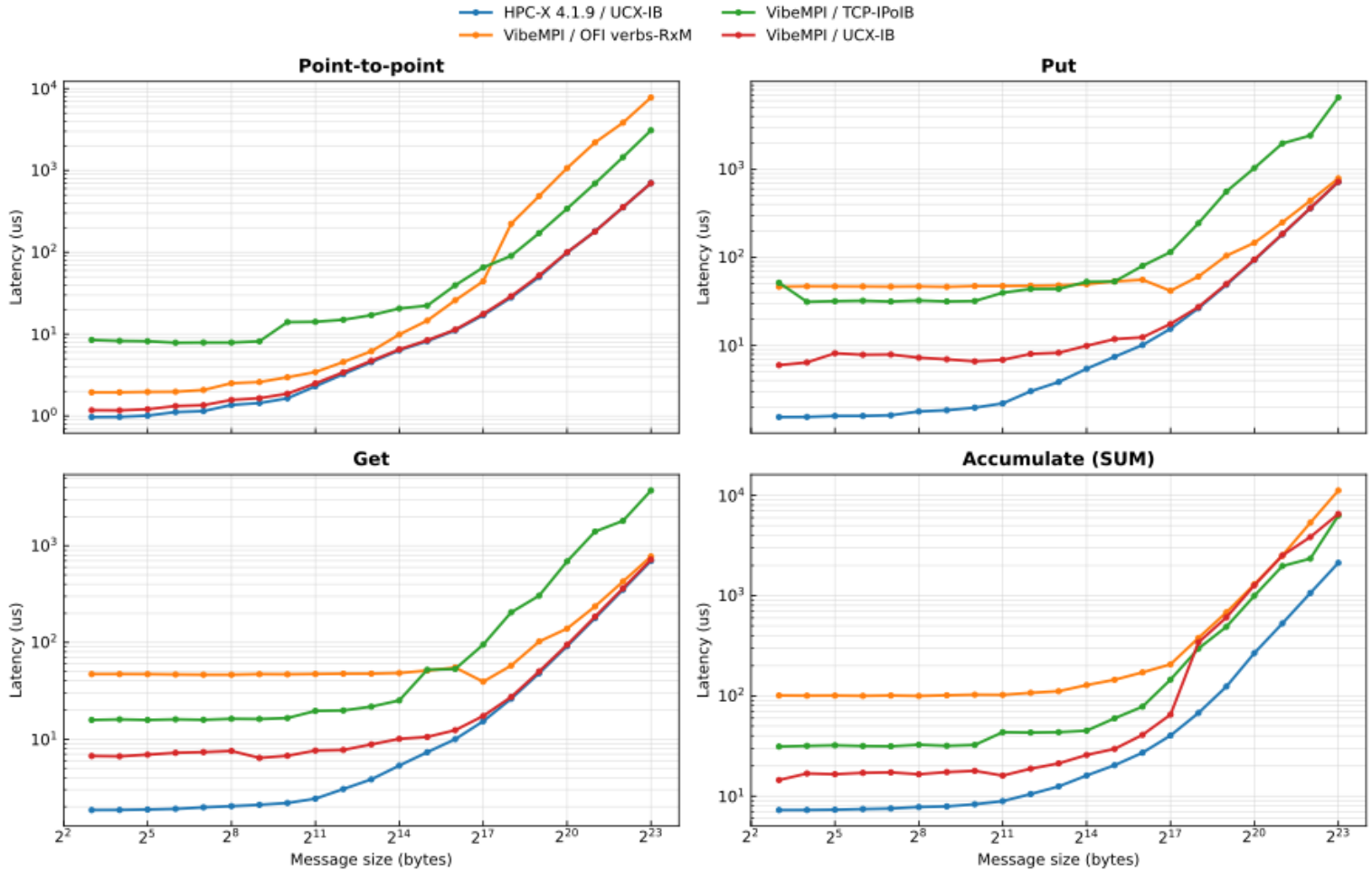
VibeMPI Features

- Full MPI-5 support, to the extent of our testing (DPM and Sessions are brittle).
- Runtime selection of transport: SHM (CMA), TCP, MUX (2), OFI, UCX.
- Matching algorithms: linear, bucketed, sharded (include no-wildcards optimization).
- RMA has both AM and native (OFI/UCX) paths. AM *requires* comm threads.
- All standard collective algorithms for barrier..scan, including Sylvain's PAT allgather, Biros' HykSort alltoallv, and other uncommon ones.
- All options (>100) controllable by env var, MPI_T or MPI info.

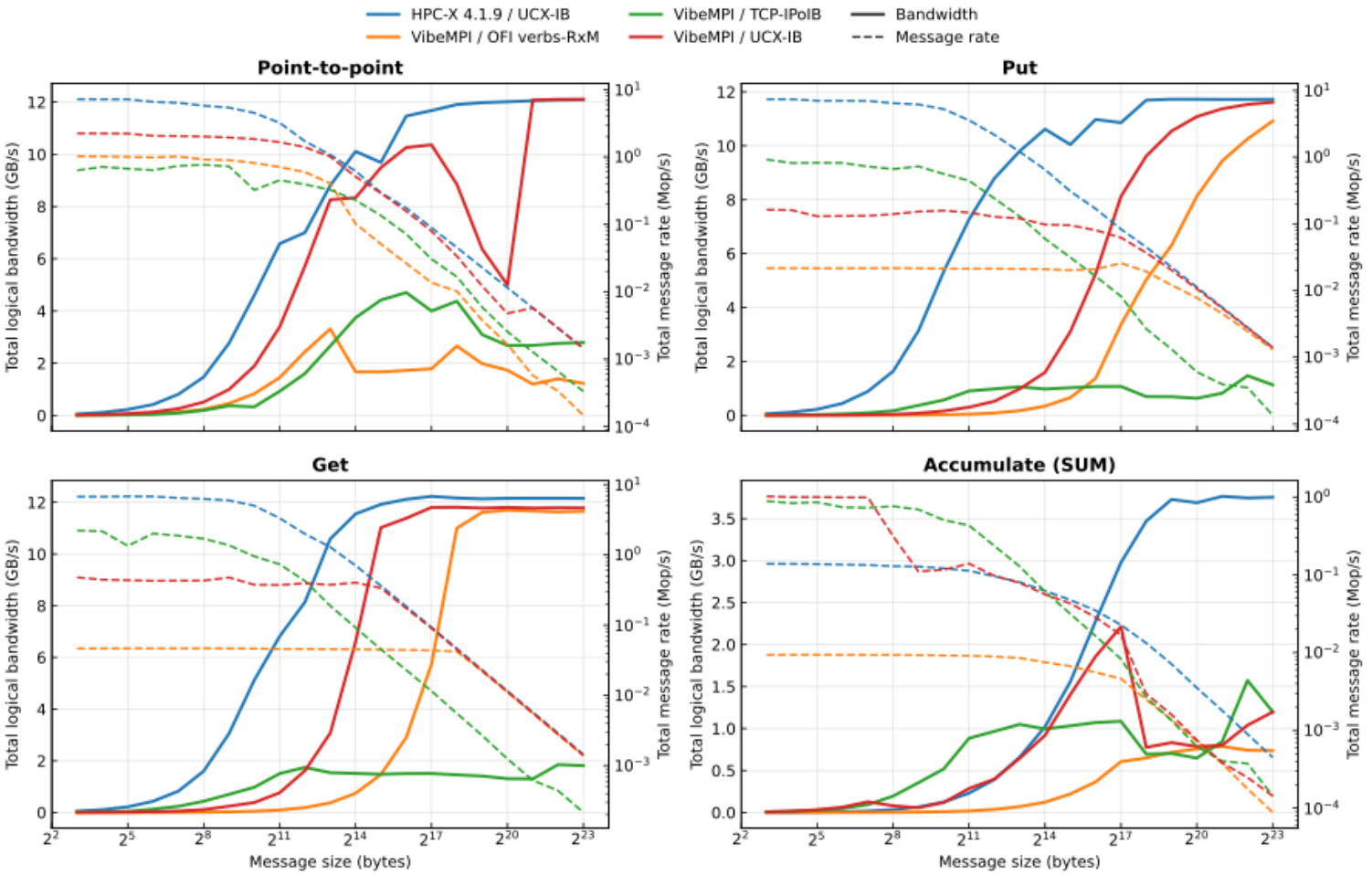
VibeMPI Testing

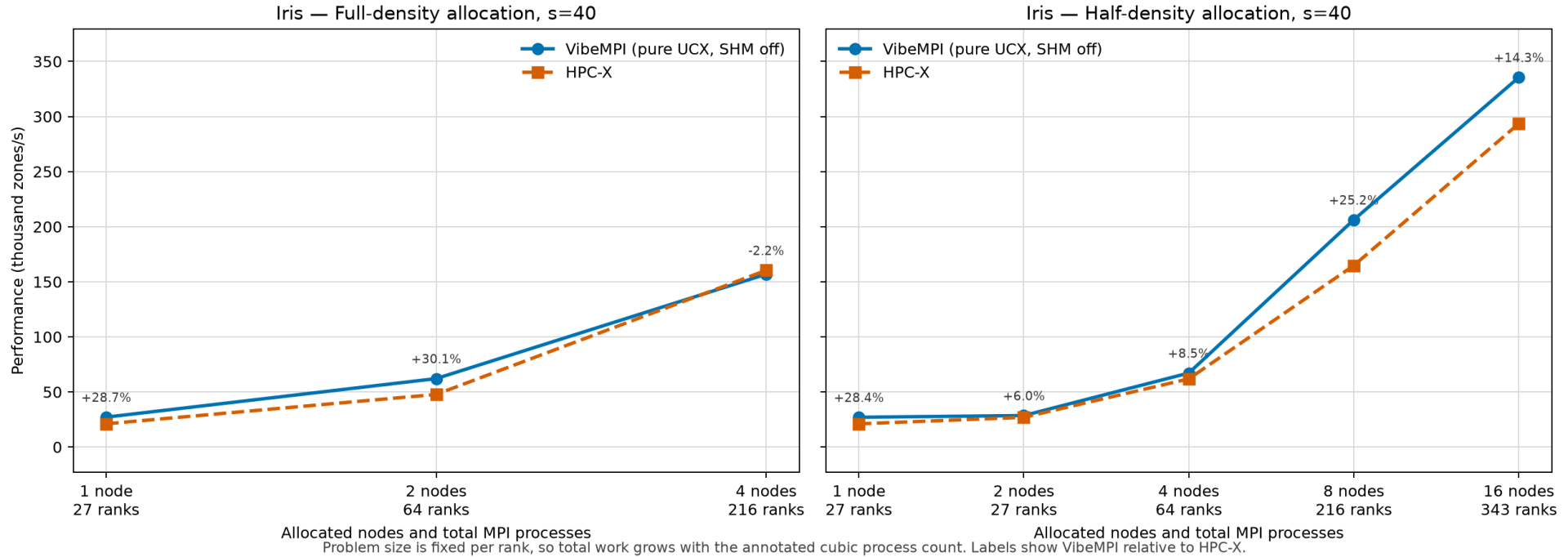
- MPICH (490), OMPI/IBM (449), Intel (137), OSU (78), mpi4py (33), OpenHPCA (26), ARMCI-MPI (24), BigMPI (16), PRK (14), Sandia (12), MTS (8), OTP (84), other (23).
- Own: collective (251), conformance (138), core (66), RMA (65), launcher (6), ULFM (3), accuracy/profiling/MPIX/MPI_T (9).
- The standard sweep is 1930 tests, with a handful of XFAIL/WONTFIX that look for implementation-specific behavior (e.g. MPICH error checking of collective arguments).
- AI ran GCOV analysis and test coverage is ~99% in most places. This required dozens of new tests.
- ARMCI-MPI is the best-known RMA compliance test, but we wrote numerous new RMA tests to cover more exotic scenarios required by the standard.
- *The VibeMPI test suite will be released as a separate project for others to use*.

MPI inter-node latency on Iris



MPI inter-node throughput on Iris




LULESH node/process scaling
Iris — Full-density allocation, s=40
Iris — Half-density allocation, s=40
VibeMPI (pure UCX, SHM off)
HPC-X
Performance (thousand zones/s)
0
50
100
150
200
250
300
350
+28.7%
+30.1%
-2.2%
+28.4%
+6.0%
+8.5%
+25.2%
+14.3%
1 node
27 ranks
2 nodes
64 ranks
4 nodes
216 ranks
2 nodes
27 ranks
4 nodes
64 ranks
8 nodes
216 ranks
16 nodes
343 ranks
Allocated nodes and total MPI processes
Allocated nodes and total MPI processes
Problem size is fixed per rank, so total work grows with the annotated cubic process count. Labels show VibeMPI relative to HPC-X.

Agentic Development Lessons

- The CXI provider for OFI is *very different* from the IB or TCP ones. One cannot rely on a generic OFI "skill" – rather one needs network-specific details in the skill.
- Code that worked reliably on IB clusters was completely broken on SS11 and has taken days to debug.
- The CXI port was relatively vibe-coded until I could see the agent was failing to understand basic concepts. I did a bottom-up analysis of the OFI provider behavior to figure out how Slingshot works.
- I will revisit VibeMPI UCX support now that I've learned a better method for porting VibeMPI to HPC networks.
- *VibeMPI will likely be open-sourced and retired this year. VibeMPI will be shared with the HPC research community as a case study.*

Agentic MPI Testing

Agent-driven testing, root-causing, MCVE-ing, reporting, and sometimes fixing of MPI libraries…

- https://github.com/pmodels/mpich/issues/7890 MPICH OFI
- https://github.com/pmodels/mpich/issues/7886 MPICH UCX
- https://github.com/pmodels/mpich/issues/7896 MPICH UCX
  (all 3 MPICH bugs were fixed by Hui within a week)

- https://github.com/open-mpi/ompi/issues/11391 OMPI UCX
  FIX: https://github.com/open-mpi/ompi/pull/14208
- https://github.com/open-mpi/ompi/issues/14173 OMPI4 UCX
  FIX: use OMPI5
- https://github.com/open-mpi/ompi/issues/14175 OMPI UCX
  FIX: https://github.com/open-mpi/ompi/pull/14207
- https://github.com/open-mpi/ompi/issues/14181 OMPI UCX
  FIX: https://github.com/open-mpi/ompi/pull/14207
- https://github.com/open-mpi/ompi/issues/14192 OMPI UCX
  FIX: https://github.com/open-mpi/ompi/pull/14205
- https://github.com/open-mpi/ompi/issues/14203 OMPI OFI
  FIX: https://github.com/open-mpi/ompi/pull/14204

All of this was done in the course of ARMCI-MPI release QA in less than a week.

What does this mean for MPI development?

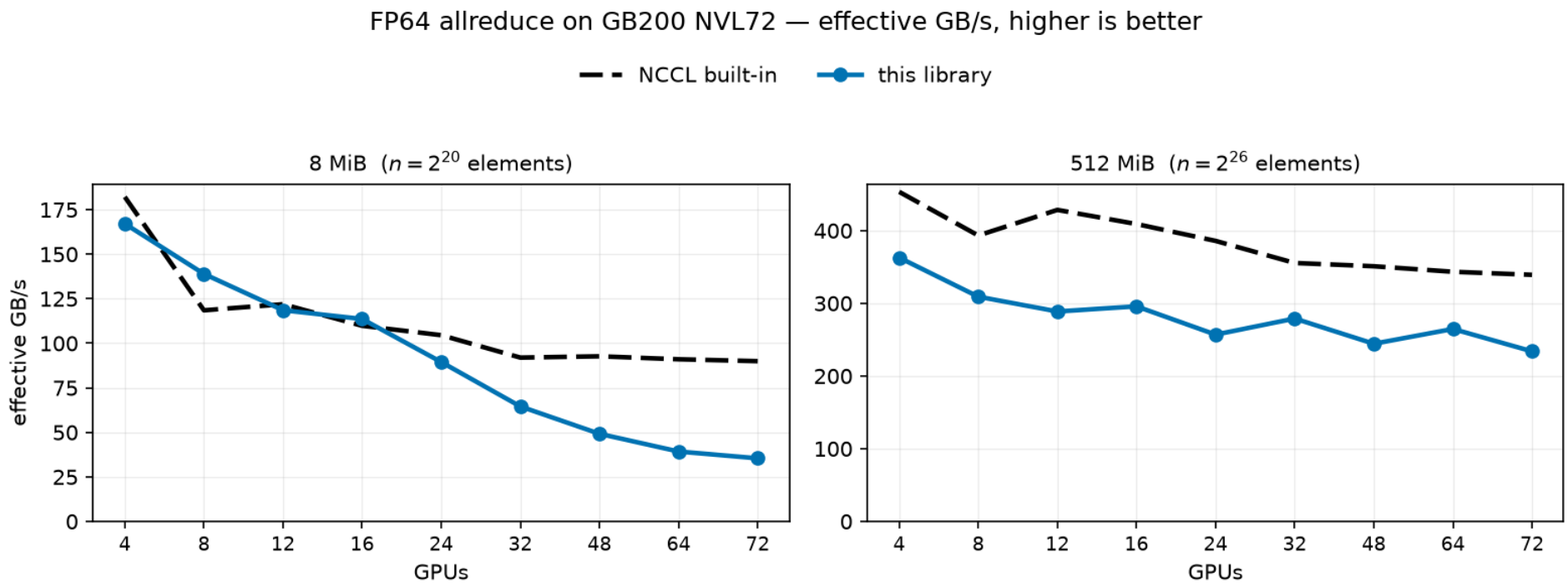
- VibeMPI was possible because we have both a high-quality, anti-error reference implementation (MPICH) and an open-source, highly tuned implementation (OMPI).
- Neither the spec nor all the tests in existence prior to this project were sufficient to produce a correct implementation.
- A correct working implementation of MPI over OFI on system 1 (IB) is not a correct working implementation of MPI over OFI on system 2 (Slingshot).
- AIDD makes it easier to implement the latest features of the standard (no excuse to not have full MPI-5 everywhere in 2027).
- AIDD works for QA on existing implementations. Fix the bugs yourself (in OSS) or demand more from your vendors.
- AIDD makes experts substantially more productive. It does not create expertise.

NCCL Libraries

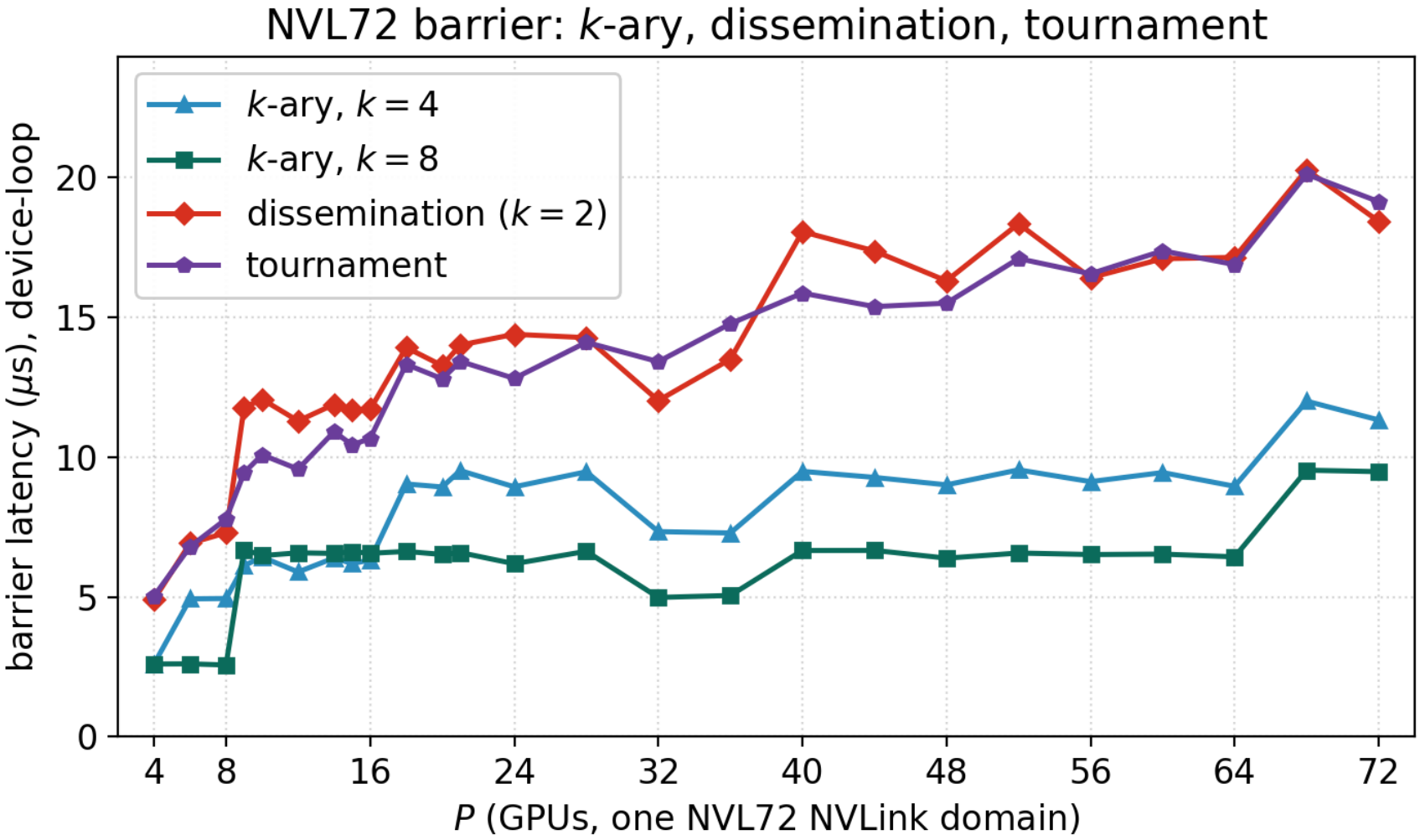
NCCL Development

- Tell the agent to read all the NCCL docs and source code at the beginning of the project; once you have one good project, tell AI to learn from it in the next one.
- Wait for the agent to produce correct code that performs terribly, then tell it to use more than one SM.
- Agents can attach NCU and do substantial kernel optimization, including register/occupancy optimizations.
- AIDD works for NCCL device-initiated communication for both NVLINK and networking (GIN), despite very few published examples. It took 1-2 days to figure it out.

NCCL NVL Reductions



Disclaimer: This is not a marketing number.


NVL72 barrier: k-ary, dissemination, tournament
k-ary, k = 4
k-ary, k = 8
dissemination (k = 2)
tournament
barrier latency (μs), device-loop
0
5
10
15
20
4
8
16
24
32
40
48
56
64
72
P (GPUs, one NVL72 NVLink domain)

Summary

Outlook for Agentic Development

- AI-driven development has progressed rapidly, with massive increase in utility around Q2 2026.
- AI is fantastic at menial tasks like Git (“rebase and fix conflicts”) and CMake. It can drive GDB, Valgrind and GCOV workflows completely autonomously. It understands Slurm.
- AI can script almost anything and generate diagrams and plots with matplotlib.
- AI will follow your instructions, even when they are bad.
- AI will be pathologically stupid mistakes, such as allocating unlimited buffering in MPI and generating obvious deadlocks.
- I rate AI at 100-1000x productivity for some tasks, and never less than 10x.

To build a building, you need an architect, an engineer, a construction manager, and a lot of carpenters.

An architect can’t build a building without carpenters, but if you have an thousand carpenters and no architect, you won’t get a building either.

AI is the thousand carpenters. You are the architect the engineer and the construction manager.

The end