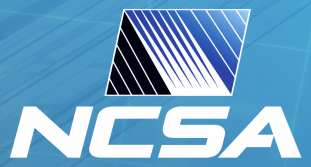
Thinking About Parallelism and Programming

William Gropp
wgropp.cs.illinois.edu



National Center for Supercomputing Applications
University of Illinois at Urbana–Champaign

Embrace the community

- Ken is best known for his contributions to parallel compilers and improving the productivity of parallel programming
- Less well known is that MPI started because of Ken
- He organized a Workshop on Standardization of Message Passing Libraries (Williamsburg, Summer 1992)
  - To find out if there could be a standard so that parallel compiler work could focus on building compilers rather than adapting to each vendor's internode communication
  - We were pretty sure we knew the answer at the end – No
  - Ken knew the answer was yes, and challenged us to solve the problem
- Lesson – Find solutions that work – independent of what you think
  - *The emergence of the MPI message passing standard for parallel computing*, R Hempel and D Walker, Computer Standards & Interfaces, 21:1, May 25, 1999, p 51-62

Training students to think precisely and quantitatively about parallel computing

- Parallel programming is *hard*
- We need to be scientific about solving these problems
- We would all like parallel programming to be eaiser and more fun, but to accomplish that, we need to focus on the real problems
- And we must set a good example for our students.
  - What follows are some examples of fuzzy thinking that we, as a community, must strive to improve
  - We can do this by insisting that our students be rigorous and to follow the scientific method
  - We must remind each other to separate opinion from fact …

Quotes from "Enabling Technologies for Petaflops Computing" (MIT Press 1995)

- "**The software for the current generation of 100 GF machines is not adequate to be scaled to a TF…**"
- "The Petaflops computer is achievable available in about 20 years [2014]."
  - (estimated clock speed in 2004 — 700M
- "Software technology for MPP's must that is portable across a wide variety o can the small but important MPP sector of the computer hardware market leverage the massive investment that is being applied to commercial software for the business and commodity computer market."
- "To address the inadequate state of software productivity, there is a need to develop language systems able to integrate software components that use different paradigms and language dialects."

Except that software was adequate not only for TeraFLOP but for PetaFLOP and almost certainly ExaFLOP

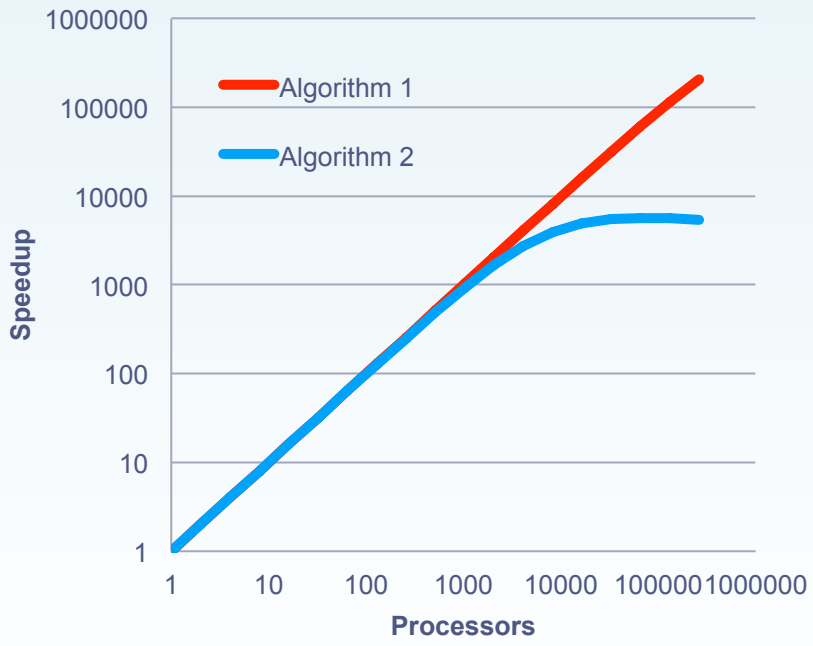
Quotes from "Enabling Technologies for Petaflops Computing" (MIT Press 1995)

- "The software for the current generation of 100 GF machines is not adequate to be scaled to a TF…"
- "**The Petaflops computer is achievable at reasonable cost with technology available in about 20 years [2014].**"
  - **(estimated clock speed in 2004 – 700MHz)**
- "Software technology for MPP… are that is portable across a wide … then can the small but important M… ket leverage the massive investme… software for the business and …
- "To address the inadequate st… eed to develop language systems able to integrate software components that use different paradigms and language dialects."

This clock speed estimate was dead on target – IBM BG/L ran at this speed, and this prediction was based on quantitative projections

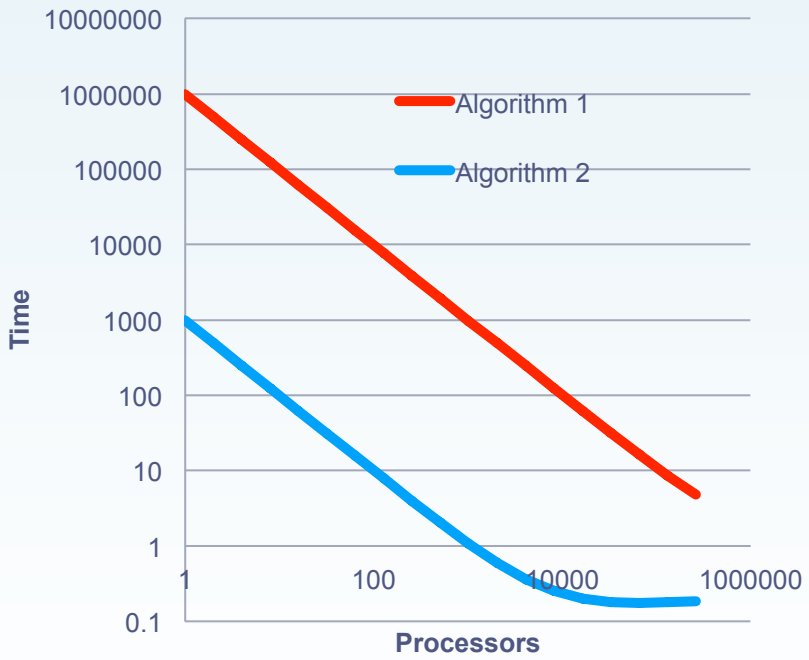
Scalability – What does it tell you?

- Should I use algorithm 1 or algorithm 2?



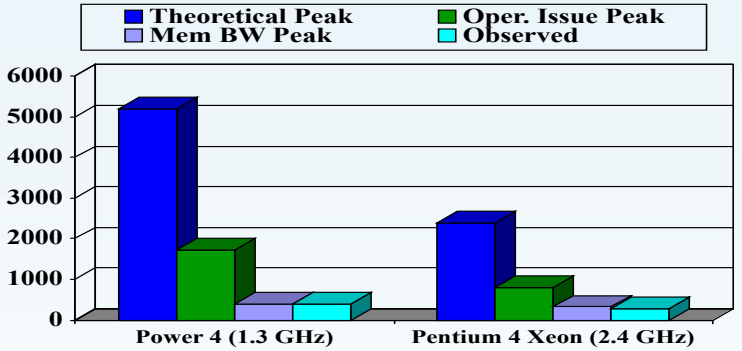
Scalability – What does it tell you?

- Should I use algorithm 1 or algorithm 2?
- Here's the same data, only showing time instead of speedup
- Its time to solution that is important
- You can always improve scalability by:
  - Decreasing per-core performance
  - Using a more computationally intensive algorithm



How should you evaluate speedup?

- When measured, you need to
  - Understand the algorithm and options
    - Some are not obvious – accurate n-body force calculations can be done in O(n), not O(n^2) time; Broadcast of n words takes O(n), not O(n log p)
  - Evaluate implementation quality
    - Can use simple quantitative performance model
    - For many apps, STREAM is appropriate

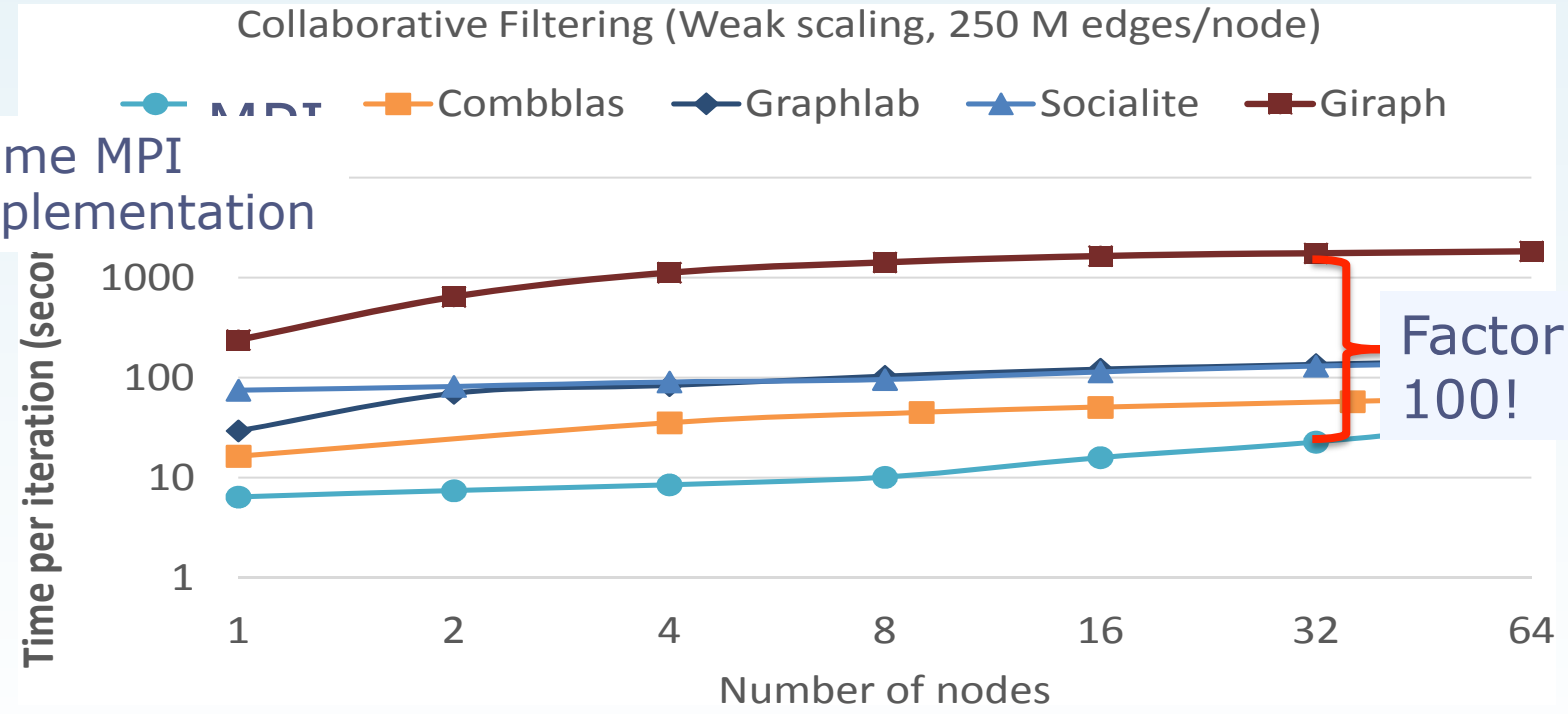


W. K. Anderson, William D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith. *Achieving high sustained performance in an unstructured mesh CFD application*, SC99 (Gordon Bell Prize)

How do you compare two programming models?

- First – are you comparing programming *models*, programming *systems*, or *implementations* of programming systems?
  - Answer – Almost always implementations
  - Implication – No paper should be accepted that claims to compare X to Y when all it does is compare an implementation of X on Z to an implementation of Y on Z
- Second – what conclusions can you draw from the comparison?
  - Example (drawn from a real vendor document)
  - Intended message: use our non-standard method – its faster!
  - Actual message: We have no clue how to implement MPI correctly

A comparison of implementations



Navigating the Maze of Graph Analytics Frameworks using Massive Graph Datasets
Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Jiwon Seo, Jongsoo Park, M. Amber Hassaan, Shubho Sengupta, Zhaoming Yin, and Pradeep Dubey

Reproducibility and significant digits

- A keystone of science is the reproducible experiment
  - Part of this is measuring and reporting only what is reproducible
- There are many challenges in making experiments reproducible
  - Complete description of the apparatus (e.g., system, compiler version/ options, source code, …)
- But some are (relatively easy)
  - Don’t report overly precise measurements
    - Or, don’t use output from %e in your paper; even %.2e is often better
  - Describe your test environment
  - Make the code and data available

What's in a name?

- In science, rigor and precision are important
- Example:
  - "Programming exascale systems requires moving beyond the BSP programming model; we need to replace MPI with <some new thing>."
- What is wrong with this statement?
  - "…exascale systems requires moving beyond BSP"
  - This is a hypothesis. How would you test or demonstrate it?
  - But there is a more subtle error…

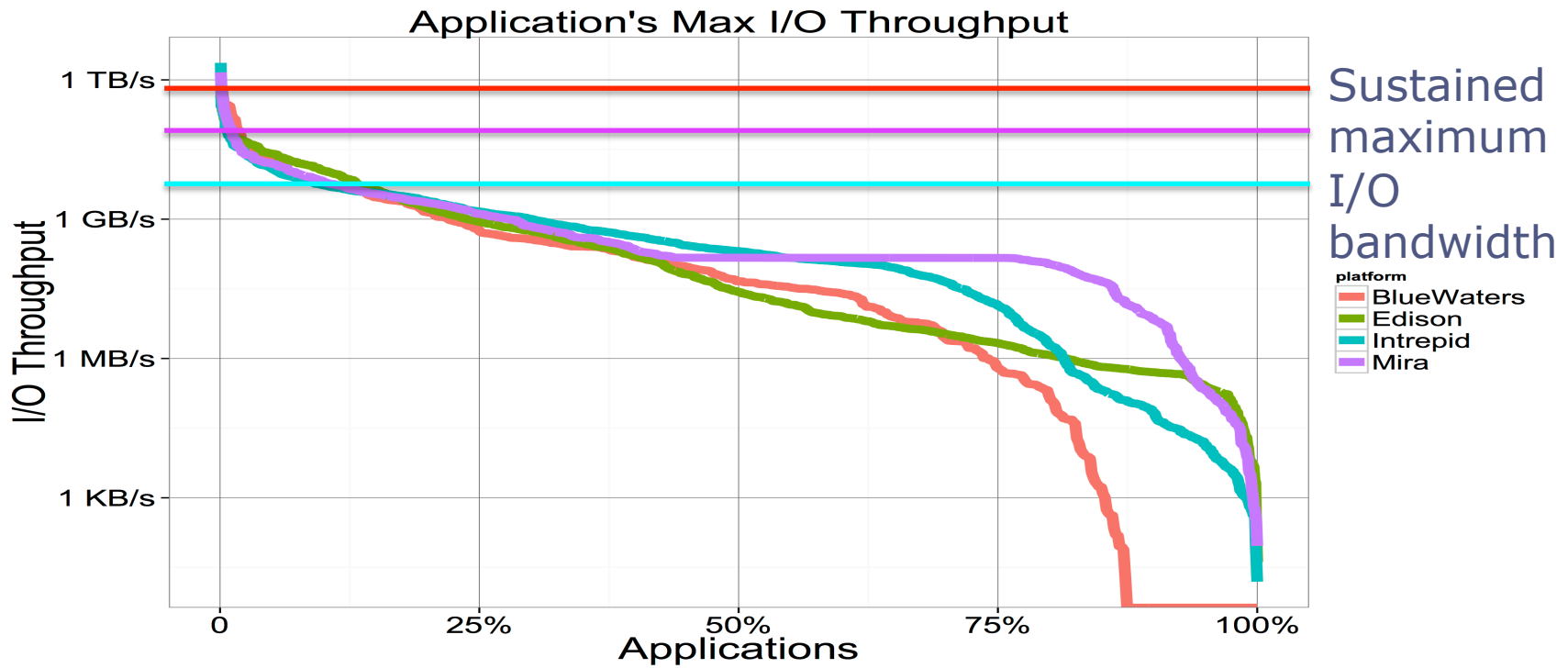
MPI is not a BSP system

- BSP = Bulk Synchronous Programming
  - Programmers **like** the BSP model, adopting it even when not necessary (see FIB)
  - Unlike most programming models, *designed* with a performance model to encourage *quantitative* design in programs
- MPI makes it easy to emulate a BSP system
  - Rich set of collectives, barriers, blocking operations
- MPI (even MPI-1) sufficient for dynamic adaptive programming
  - The main issues are performance and “progress”
  - Improving implementations and better HW support for integrated CPU/NIC coordination more effective at supporting need

Understand what you ask for

- Current I/O performance is often appallingly poor
  - Even relative to what current systems can achieve
  - Part of the problem is the I/O interface semantics
- Many applications need to rethink their approach to I/O
  - Not sufficient to “fix” current I/O implementations
- HPC Centers have been complicit in causing this problem
  - By asking users the wrong question
  - By using their response as an excuse to keep doing the same thing
  - What is that question that causes so much trouble?
    - Do you want/need POSIX I/O?

Just how bad Is current I/O performance?



"*A Multiplatform Study of I/O Behavior on Petascale Supercomputers*," Huong Luu, Marianne Winslett, William Gropp, Robert Ross, Philip Carns, Kevin Harms, Prabhat, Suren Byna, and Yushu Yao, proceedings of HPDC'15.

What are some of the problems?

- POSIX I/O has a strong consistency model
  - Extremely hard to cache effectively
  - Applications need to transfer block-aligned and sized data to achieve performance
  - Complexity adds to fragility of file system, the major cause of failures on large scale HPC systems
- Files as I/O objects add metadata “choke points”
  - Serialize operations, even with “independent” files
- Burst buffers will not fix these problems – must change the semantics of the operations

Other communities have matched their data systems to their needs

- “Big Data” file systems have very different consistency models and metadata structures, designed for their application needs
  - Why doesn’t HPC?
    - There have been some efforts, such as PVFS, but the requirement for POSIX has held up progress

Why is the question “wrong”?

- To almost all application developers, asking if they need POSIX I/O means (to them)
  - Do you need open/seek/read/write/close?
- And if they answer “no”, the implication is
  - They will need to rewrite their application and reformat their files
- With this interpretation, **no-one** would ever answer “no” to “Do you need POSIX I/O?”
  - Questions to which only one answer is reasonable don’t tell you anything

No science application code needs POSIX I/O semantics

- Many are single reader or single writer
  - Eventual consistency is fine
- Some are disjoint reader or writer
  - Eventual consistency is fine, but must handle non-block-aligned writes
- Some applications use the file system as a simple data base
  - Use a data base – we know how to make these fast and reliable
- Some applications use the file system to implement interprocess mutex
  - Use a mutex service – even MPI point-to-point
- A few use the file system as a bulletin board
  - Most likely better off using RDMA
  - Only need release or eventual consistency
- *Correct* Fortran codes do not require POSIX in any form
  - Standard requires unique open, enabling correct and aggressive client and/or server-side caching
- MPI-IO would be better off without POSIX

Where's the hope?

- I firmly believe that a rigorous, scientific approach is the only way to solve the great challenges facing us in making better, more productive use of parallel computing
- And look where it has gotten us already
  - Computing power in a single system has increased one ***billion*** fold over my career. Data capacity and networking bandwidths have seen similar increases
  - We have been able to program these systems
    - Often the real challenge is in single core or single node performance
- We can accelerate this progress by (training our students in) challenging conventional wisdom and being rigorous in applying the scientific method

Thanks!

- For funding from
  - National Science Foundation
  - Department of Energy
  - ExxonMobile and JumpLabs
  - State of Illinois
- My co-workers at Yale, Argonne, and Illinois
- My colleagues around the world
- My students
- And especially Ken Kennedy, for his contributions and for the example he set as a scholar and gentleman