Parallel Programming with MPI on Clusters

Rusty Lusk
Mathematics and Computer Science Division
Argonne National Laboratory

(The rest of our group: Bill Gropp, Rob Ross, David Ashton, Brian Toonen, Anthony Chan)

Outline

- Clusters are a significant component of high-performance computing. (Duh!)
- MPI is a significant component of the programming and execution environment on clusters.
- This talk touches on three components of the MPI universe:
  - The MPI Standard
    - And why it has been “successful”
  - Implementation issues and status
    - With a little extra on MPICH
  - Non-MPI software that interacts with MPI implementations
    - Tools and environments
- An example MPI application
  - Illustrates several points, excuse to show pretty pictures

What is MPI?

- A *message-passing library specification*
  - extended message-passing model
  - not a language or compiler specification
  - not a specific implementation or product
- For parallel computers, clusters, heterogeneous networks
- Full-featured
- Designed to provide access to advanced parallel hardware for
  - end users
  - library writers
  - tool developers

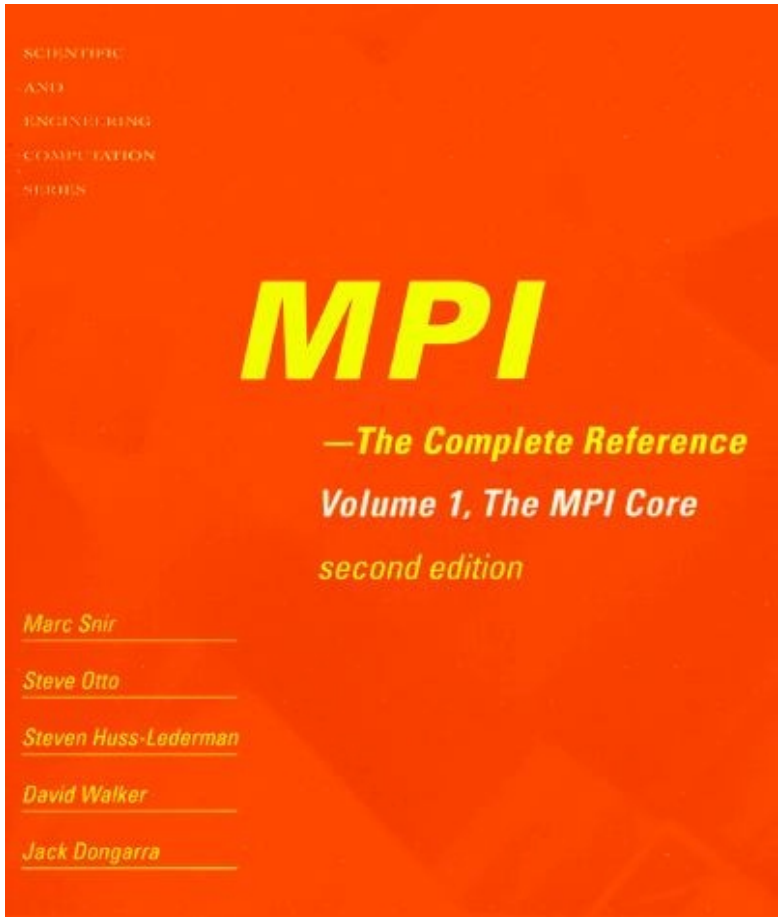
Where Did MPI Come From?

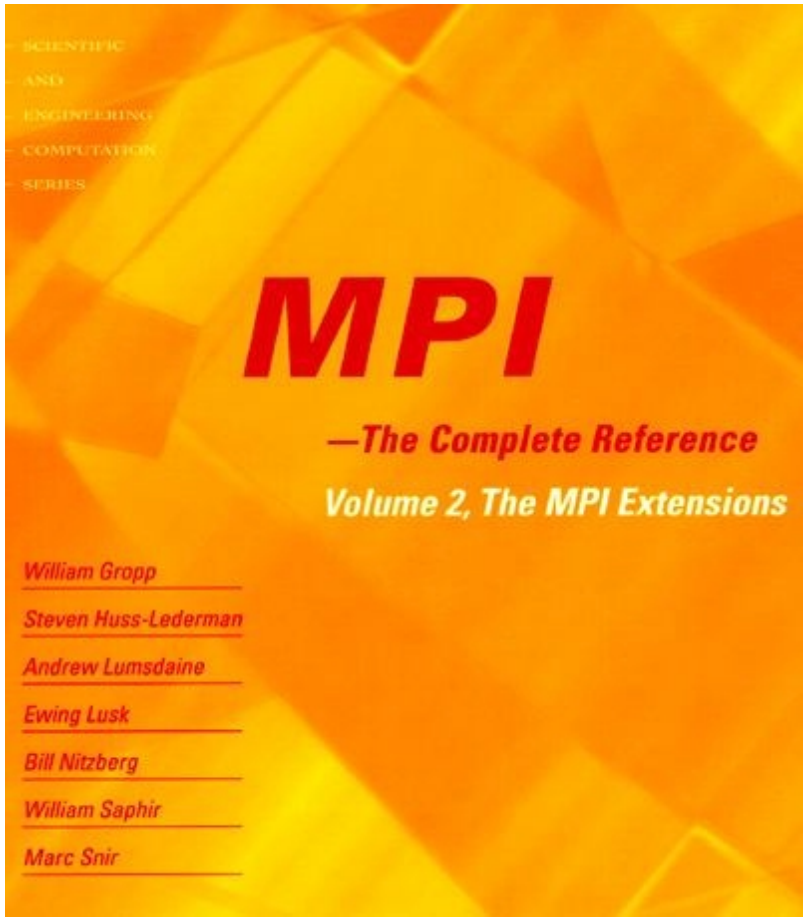
- Early vendor systems (NX, EUI, CMMD) were not portable.
- Early portable systems (PVM, p4, TCGMSG, Chameleon) were mainly research efforts.
  - Did not address the full spectrum of message-passing issues
  - Lacked vendor support
  - Were not implemented at the most efficient level
- The MPI Forum organized in 1992 with broad participation by vendors, library writers, and end users.
- MPI Standard (1.0) released June, 1994; many implementation efforts.
- MPI-2 Standard (1.2 and 2.0) released July, 1997.
- MPI-2.1 being defined now to remove errors and ambiguities.

MPI Sources

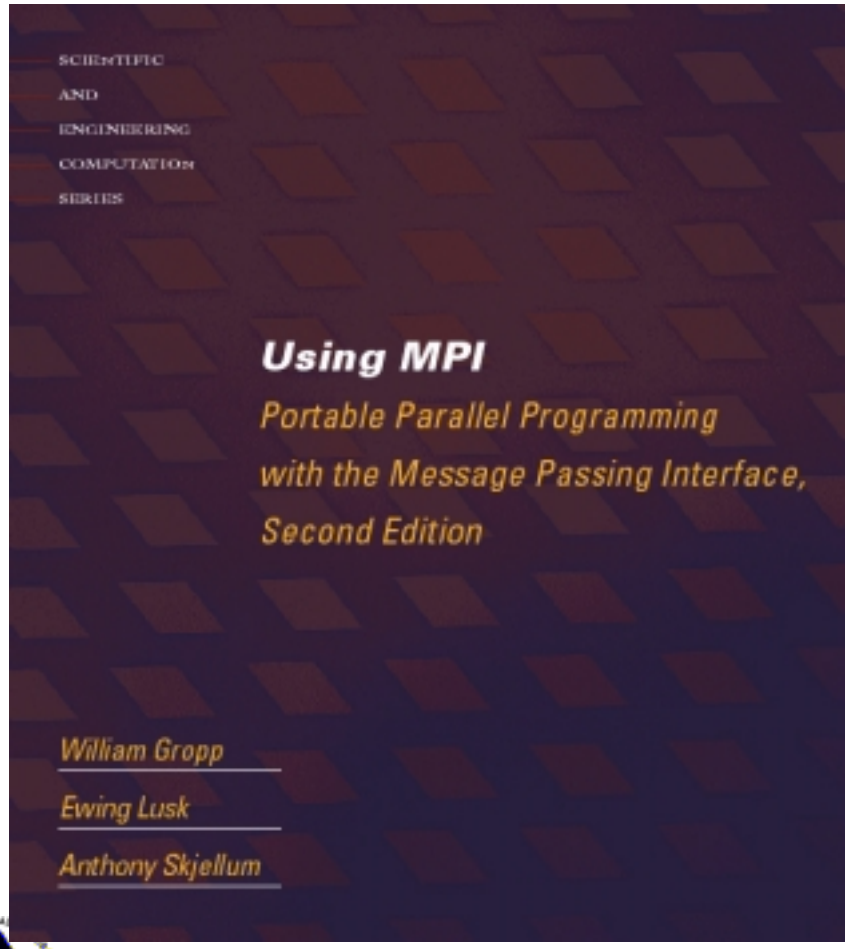
- The Standard itself:
  - at **`http://www.mpi-forum.org`**
  - All MPI official releases, in both postscript and HTML
- Books on MPI and MPI-2:
  - *MPI: The Complete Reference, volumes 1 and 2*, MIT Press, 1999.
  - *Using MPI: Portable Parallel Programming with the Message-Passing Interface (2nd edition)*, by Gropp, Lusk, and Skjellum, MIT Press, 1999.
  - *Using MPI-2: Extending the Message-Passing Interface*, by Gropp, Lusk, and Thakur, MIT Press, 1999
- Other information on Web:
  - at **`http://www.mcs.anl.gov/mpi`**
  - pointers to lots of stuff, including other talks and tutorials, a FAQ, other people's MPI pages

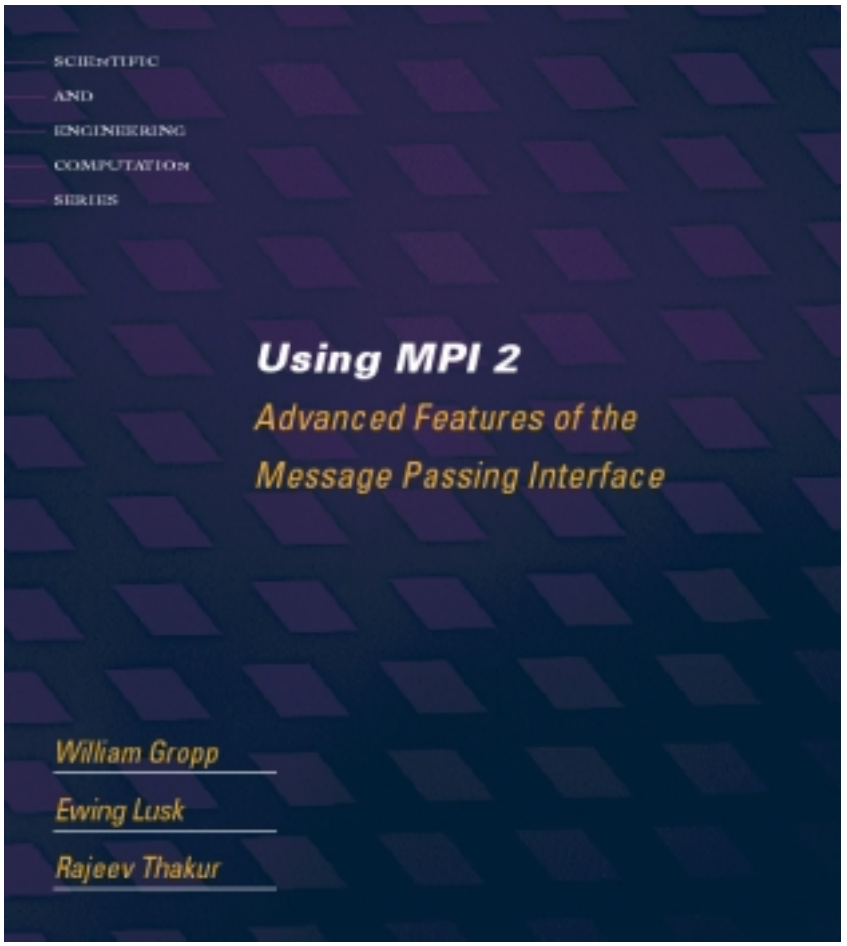
The MPI Standard Documentation





Tutorial Material on MPI, MPI-2





Why Has MPI Succeeded?

(Important to understand when contemplating alternatives)

- Open process of definition
  - All invited, but hard work required
  - All drafts and deliberations open at all times
- Portability
  - Need not lead to lowest common denominator approach
  - MPI semantics allow aggressive implementations
- Performance
  - MPI can help manage the memory hierarchy
  - Collective operations can provide scalability
  - Cooperates with optimizing compilers
- Simplicity
  - MPI-2 has 275 functions; is that a lot?
  - Can write serious MPI applications with 6 functions
  - Economy of concepts

Why Has MPI Succeeded?
(continued)

- Modularity
  - MPI supports component-based software with communicators
  - Support for libraries means some applications may contain no MPI calls
- Composability
  - MPI works with other tools (compilers, debuggers, profiliers)
  - Provides precise interactions with multithreaded programs
- Completeness
  - Any parallel algorithm can be expressed
  - Easy things are not always easy with MPI, but
  - Hard things are possible

MPI Implementation Status

- All parallel computer vendors have MPI-1, and some have complete MPI-2 implementations.
- Implementations for clusters
  - MPICH, LAM, MPI-Pro have MPI-1, parts of MPI-2 for Linux clusters
  - For Windows2000, MPICH and MPIPro
- MPICH-derived special implementations
  - Myricom's MPICH-GM (for Myrinet)
  - Globus's MPICH-G2 (for multiple MPI's)
  - Scyld's BeoMPI (for Scyld Beowulf clusters)
  - LBL's MVICH (for Linux clusters with VIA)
  - Research implementations (e.g. BIPng)
  - others

MPI Implementation Research Issues and Topics

The existence of a standard API like MPI focuses implementation research, like standard languages focus compiler research

- Datatypes
  - Packing algorithms
  - Exploiting MPI_Type_commit
- Memory motion reduction
  - Eliminating interlayer copies
  - Utilizing cache-aware data structures
- Portability and performance through lower-level communication abstractions
  - (useful even outside MPI)
- Better collective operation implementations
  - Most implementations layer on point-to-point MPI
  - Need stream-oriented methods that understand MPI datatypes

More Implementation Research Issues and Topics

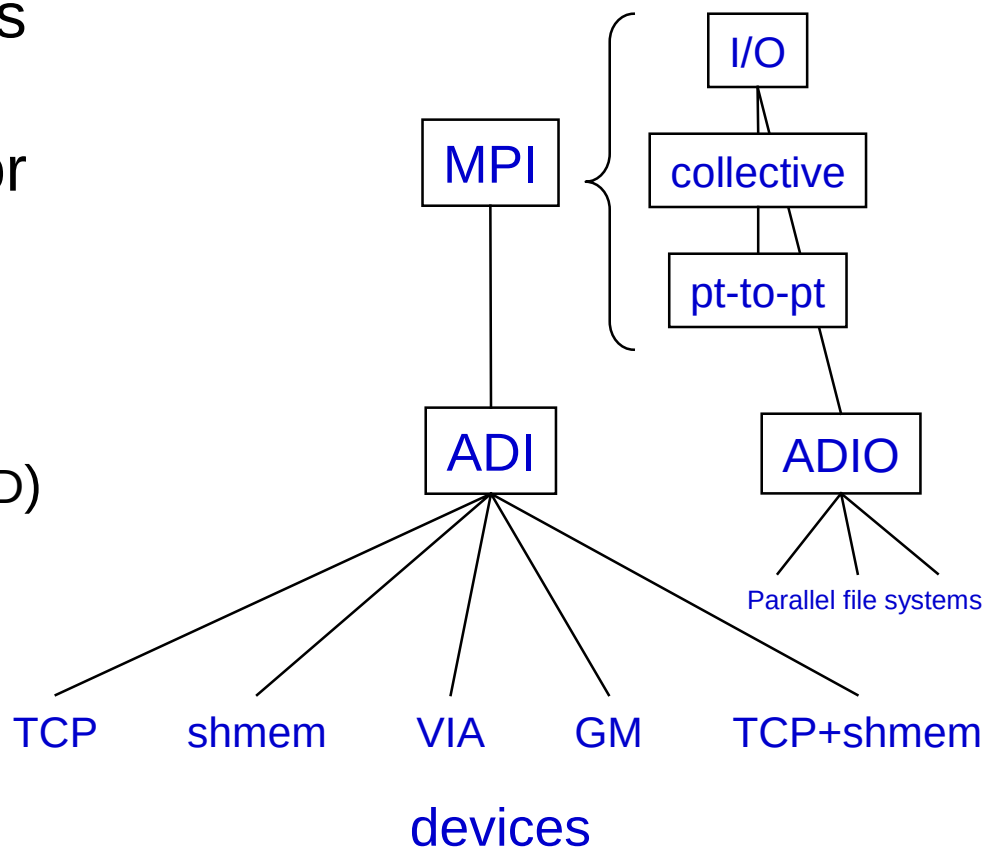
- Parallel I/O
  - Exploiting MPI collective operations
  - The abstract interface for parallel I/O
  - Tuning for cluster parallel file systems (e.g. PVFS)
- Optimizing communication algorithms and data structures for new hardware and software
  - Infiniband
  - VIA
  - What can go on the NIC?
- Fault tolerance
  - Intercommunicators can provide one approach
- Checkpointing
  - Interfaces for saving state
- Exploiting multithreading at multiple levels
- Scalable startup

The MPICH Implementation of MPI

- *As a research project:* exploring tradeoffs between performance and portability; conducting research in implementation issues.
- *As a software project:* providing a free MPI implementation on most machines; enabling vendors and others to build complete MPI implementations on their own communication services.
- MPICH 1.2.2.2 just released, with complete MPI-1, parts of MPI-2 (I/O and C++), port to Windows2000.
- Available at http://www.mcs.anl.gov/mpi/mpich

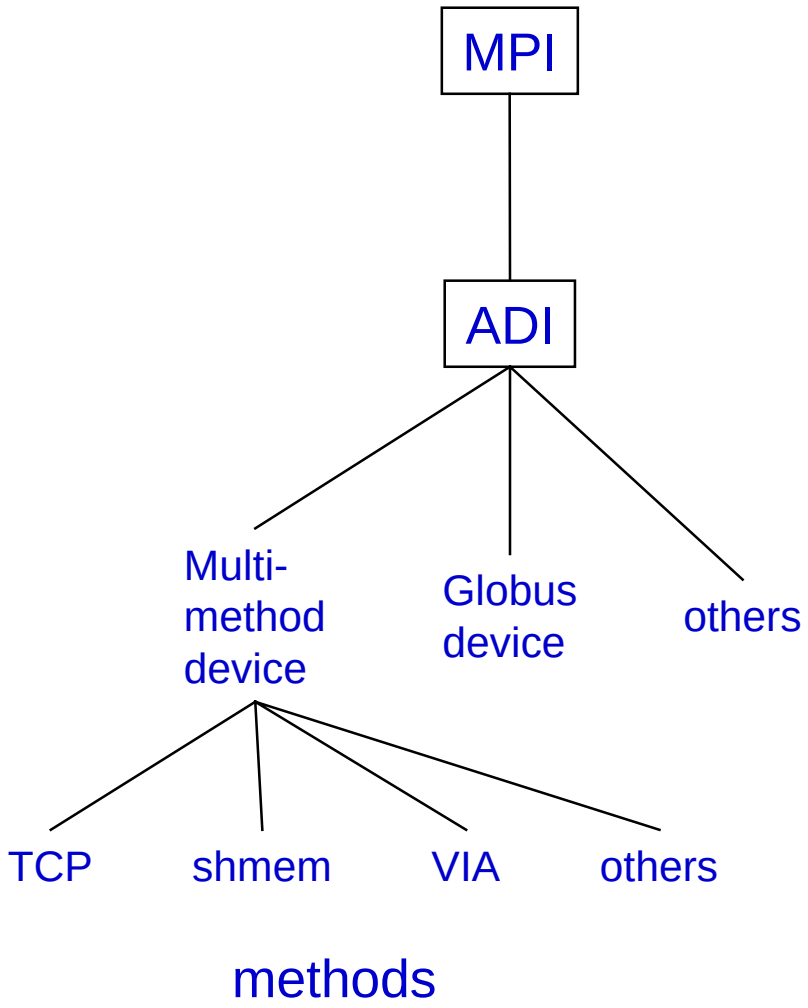
MPICH-1 Design and Status

- MPICH's architecture has encouraged its use in other projects and vendor MPI's.
- Limitations:
  - Not thread-safe (MPI_THREAD_FUNNELLED)
  - No dynamic processes
  - No RMA
- Most recent change:
  - Fast startup with MPD process manager



MPICH-2 Goals and Design

- Design goals
  - Same as before:
    - Portable and efficient
    - Modular for use by others
    - Implementation research vehicle
  - Plus:
    - All of MPI-2
    - All levels of thread support
    - Ready for next-generation hardware
    - Scalability a major goal
- Status
  - Detailed design nearly complete



MPI Works with Other Tools

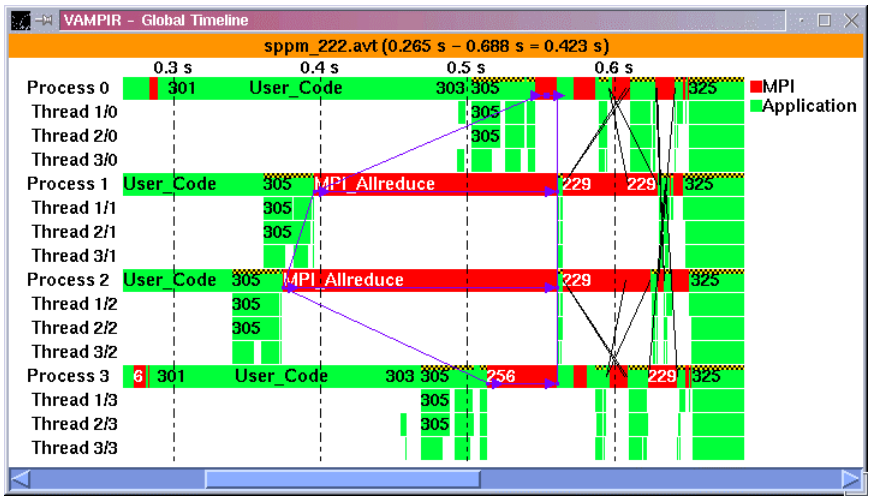
- Since it is a library, MPI applications can use latest compilers (e.g. new Intel C and Fortran compilers, choice of Fortran compilers for Linux, Windows compilers, OpenMP compilers.
- Since it supports libraries, it can be used to implement other portable software components
  - PETSc
  - ScaLAPACK
  - Global Arrays
  - Paramesh
  - HDF-5
  - Autopack
- Since it is a specification, it encourages multiple implementations and implementation research.

Interfaces Promote MPI Application Use of Tools

- The MPI Profiling Interface
  - Part of any conforming implementation
  - Encourages commercial tools (e.g., GuideView, Vampir)
  - Encourages open tools (e.g. Jumpshot, XMPI), personal tools
- The Debugger Interface (Cownie & Gropp, 1999)
  - Allows debuggers access to message queues
  - Used by TotalView
  - Implemented by MPICH and other MPI implementations
- The Process Manager Interface (Butler, Lusk, & Gropp, 2000)
  - Allows multiple Process Managers to provide startup and other services to multiple MPI implementations
  - Used by MPICH
  - Implemented by MPD Process Manager (comes with MPICH)

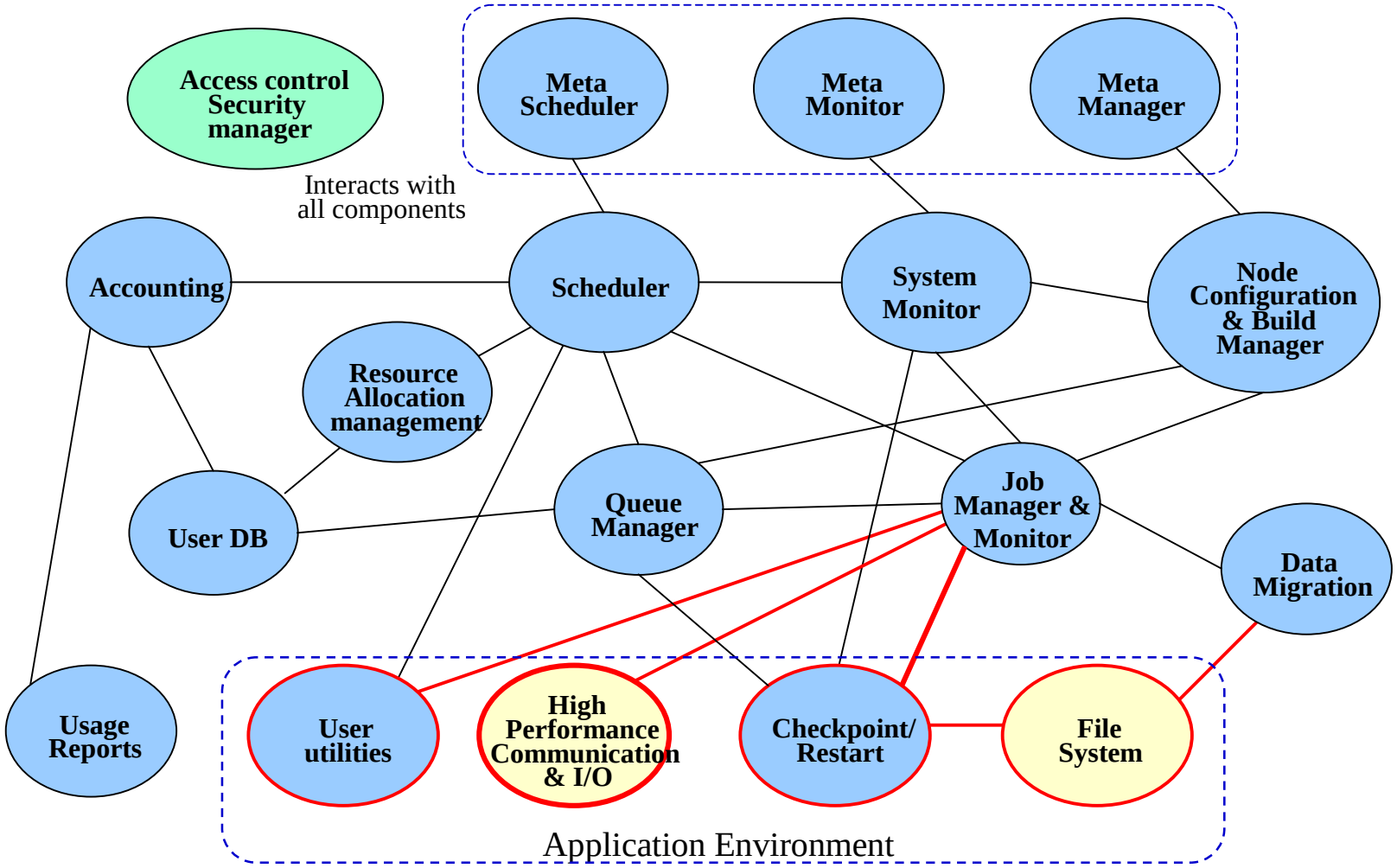
MPI and OpenMP

- MPI provides interface (MPI_Thread_init) for requesting a specific level of thread safety
  - MPI_THREAD_SINGLE – single threaded
  - MPI_THREAD_FUNNELLED – needed for loop parallelism
  - MPI_THREAD_SERIAL – needed for “single” directive
  - MPI_THREAD_MULTIPLE – needed for complete multithreading
- Thread-aware MPI tools: TotalView (Etnus) and GuideView (Pallas/Intel)



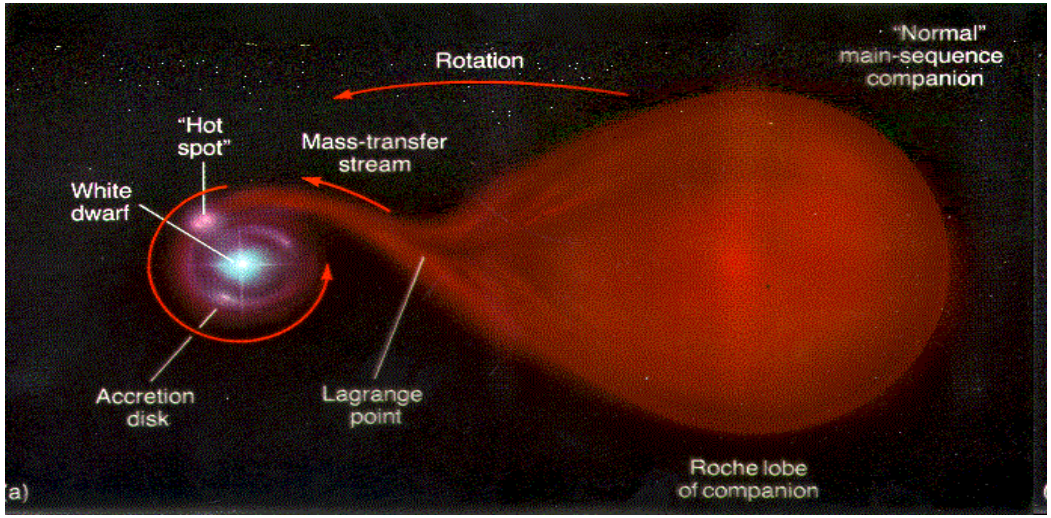
The MPI Implementation as a Component of a Cluster Environment

- A component view of cluster system software



An MPI Application

- Goal of the FLASH project: To simulate matter accumulation on the surface of compact stars, nuclear ignition of the accreted (and possibly underlying stellar) material, and the subsequent evolution of the star's interior, surface, and exterior
  - X-ray bursts (on neutron star surfaces)
  - Novae (on white dwarf surfaces)
  - Type Ia supernovae (in white dwarf interiors)

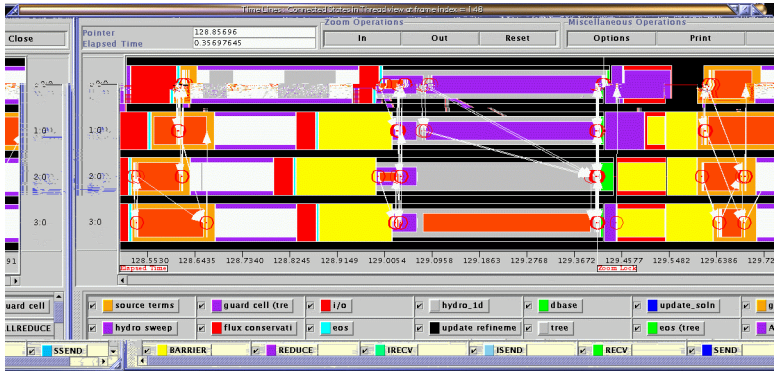


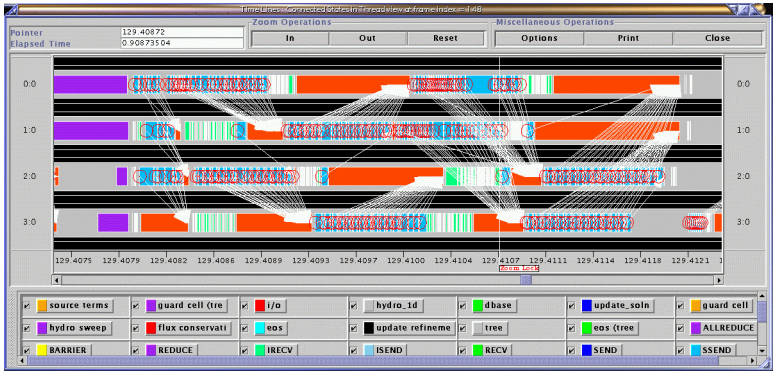
The FLASH Code

- Solves complex systems of equations for hydrodynamics and nuclear burning
- Adaptive mesh refinement on rectangular grid
- Written primarily in Fortran-90
  - A little C and Python
- Gordon Bell prize winner in 2000
- Illustrates nearly all aspects of MPI discussed here

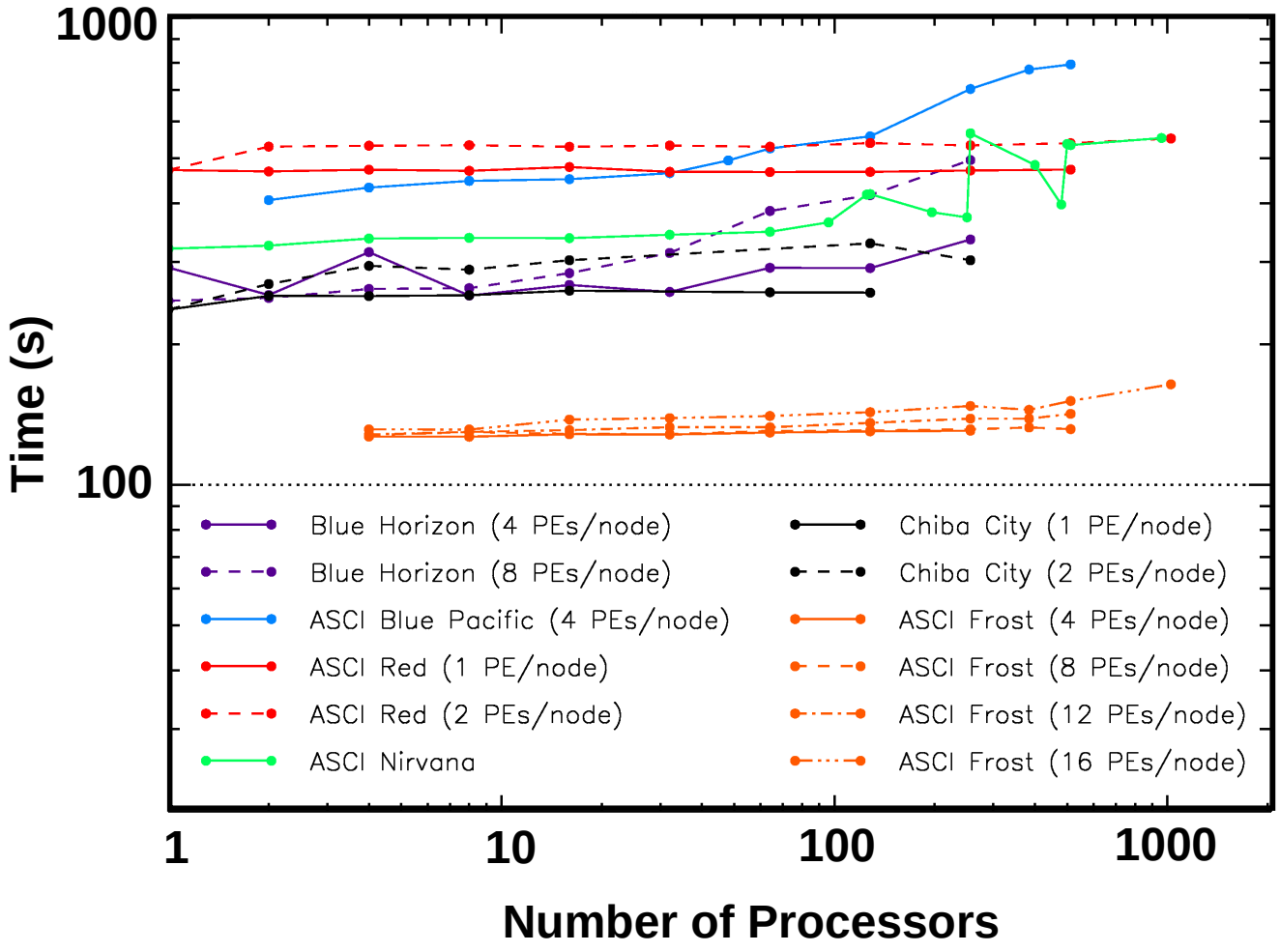
Role of MPI in FLASH

- Provides Portability and Scalability (see next slide)
- Relies heavily on MPI-based libraries
  - Uses Paramesh library for adaptive mesh refinement; Paramesh is implemented with MPI; no MPI in FLASH itself
  - Parallel I/O (for checkpointing, visualization, other purposes) done with HDF-5 library, which is implemented with MPI-IO
- Debugged with TotalView, using standard debugger interface
- Tuned with Jumpshot and Vampir, using MPI profiling interface

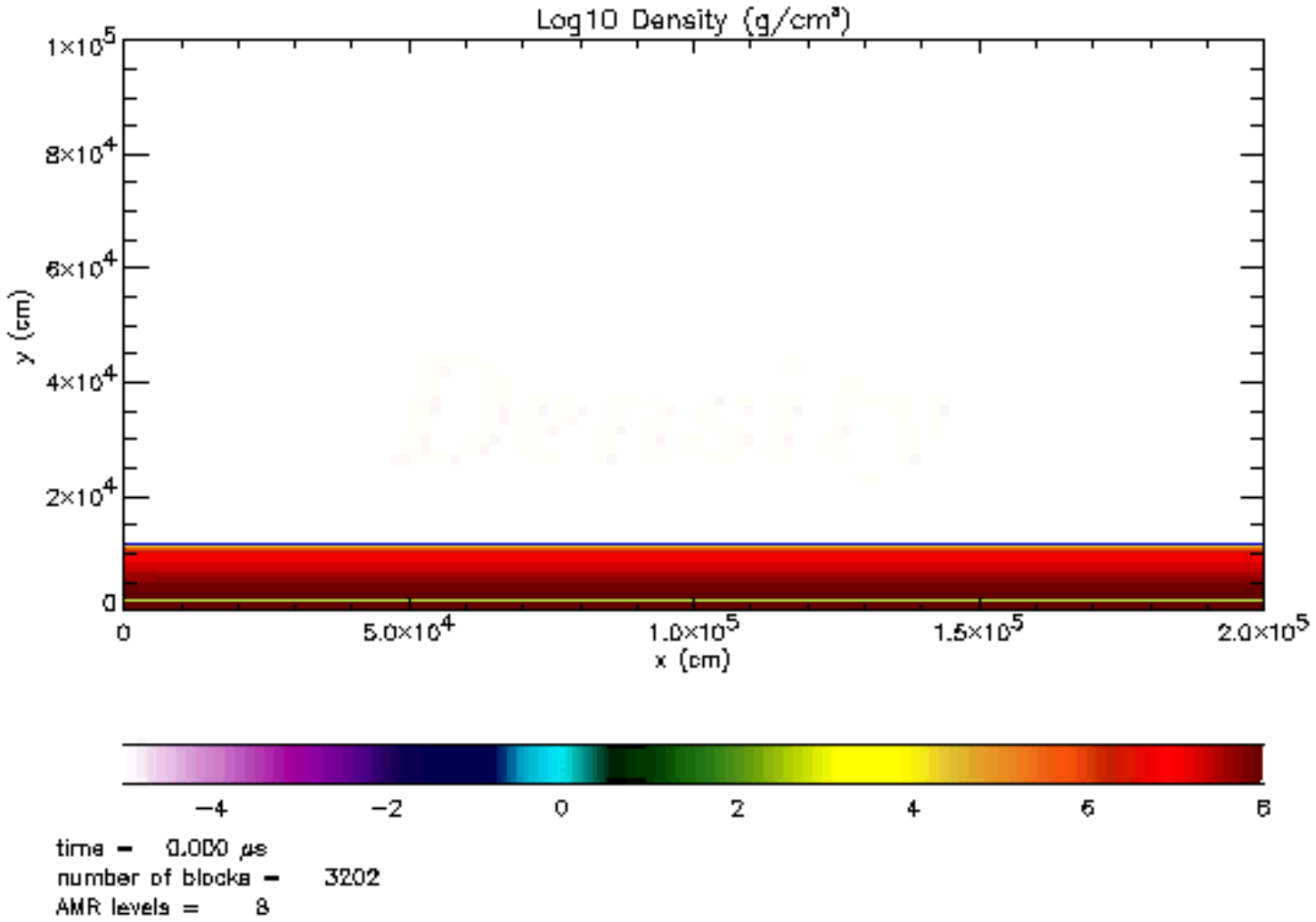




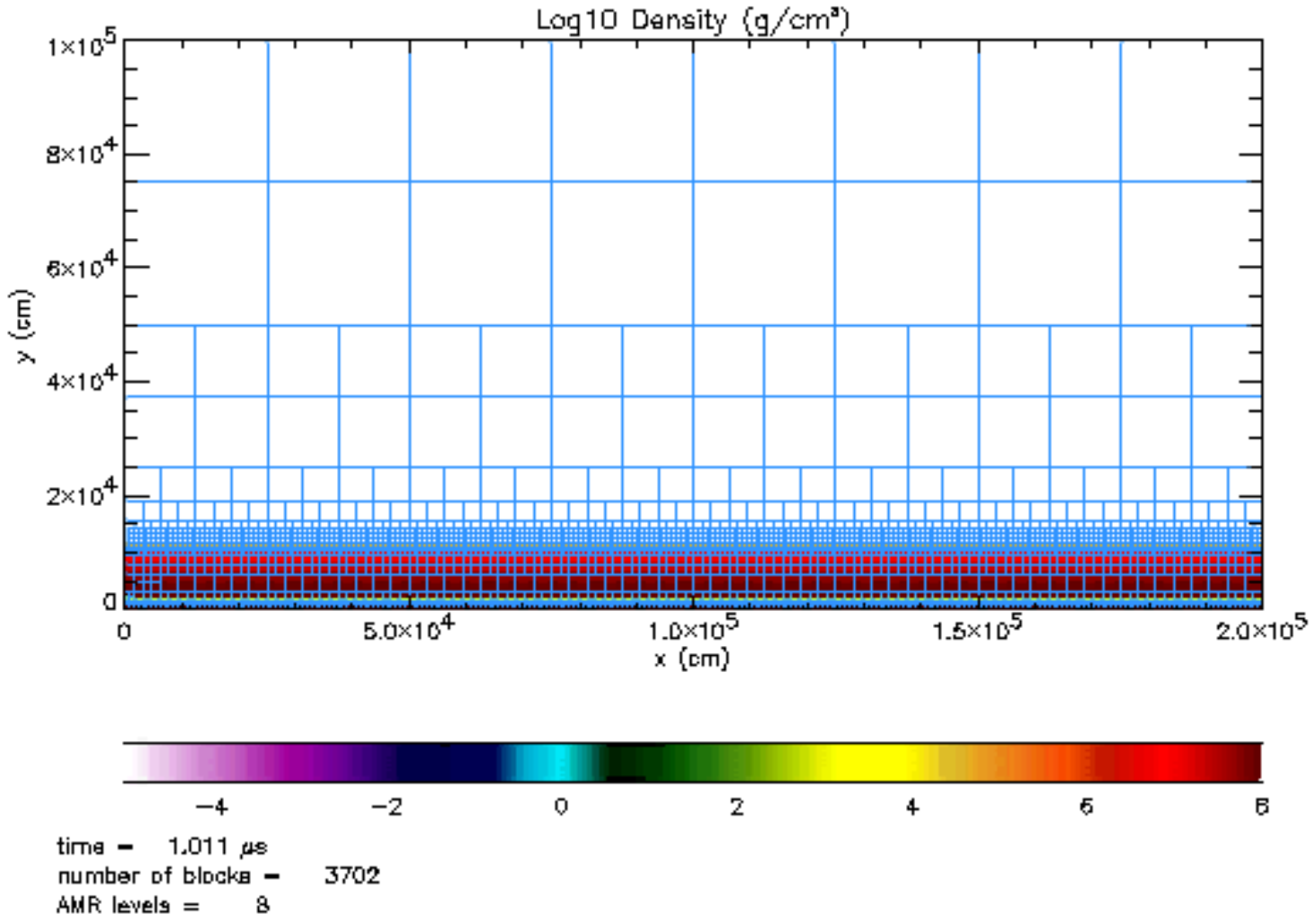
FLASH Scaling Runs



X-Ray Burst on the Surface of a Neutron Star

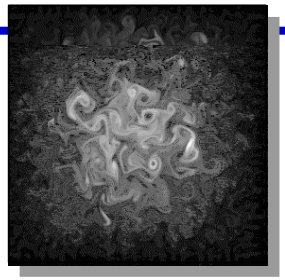


Showing the AMR Grid

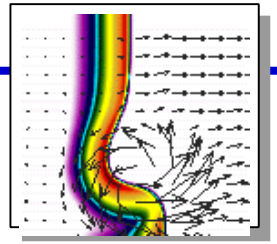


FLASH Scientific Results

- Wide range of compressibility
- Wide range of length and time scales
- Many interacting physical processes

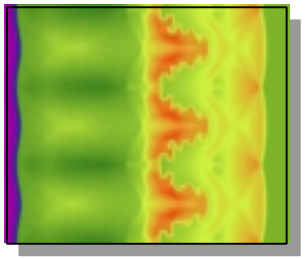


Compressible turbulence

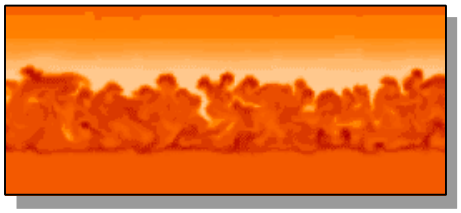


Flame-vortex interactions

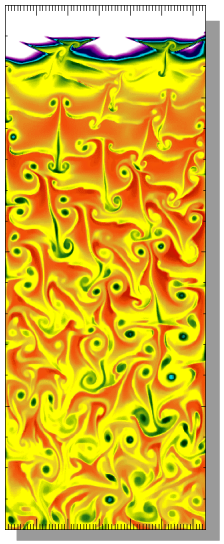
Richtmyer-Meshkov instability



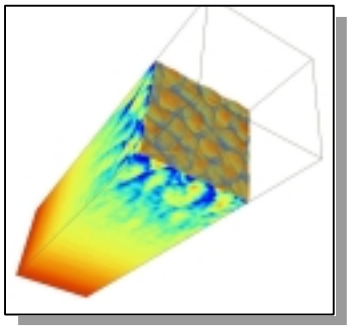
Laser-driven shock instabilities

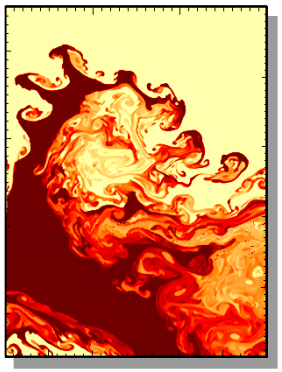


Nova outbursts on white dwarfs

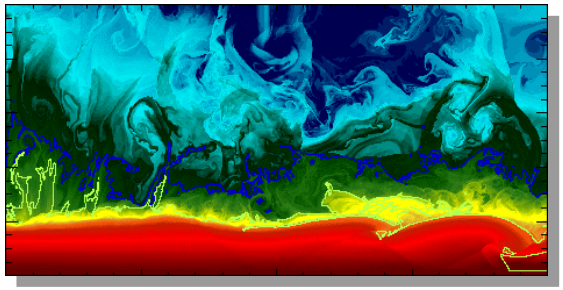


Cellular detonations





Rayleigh-Taylor instability



Helium burning on neutron stars

Future Developments in Parallel Programming: MPI and Beyond

- MPI not perfect
- Any widely-used replacement will have to share the properties that made MPI a success.
- Some directions (in decreasing order of speculativeness)
  - Improvements to MPI implementations
    - Better collective operation performance, full MPI-2
  - Improvements to the MPI definition
    - E.g., better remote memory operations
  - Continued evolution of libraries
  - Interactions with compilers
  - Further out: radically different programming models for radically different architectures.

Summary

- MPI is a successful example of a community defining, implementing, and adopting a standard programming methodology.
- It happened because of the open MPI process, the MPI design itself, and early implementation work.
- MPI research continues to refine implementations on modern platforms, and this is the “main road” ahead.
- Tools that work with MPI programs are thus a good investment.
- MPI provides portability and performance for complex applications on a variety of architectures.