HPC Software: Lessons from MPI

William D. Gropp
Mathematics and Computer Science
www.mcs.anl.gov/~gropp
Fun3d Performance

Used mixed MPI/SMP model

Performance close to “achievable peak” based on memory bandwidth
FUN3d Characteristics

- Tetrahedral vertex-centered unstructured grid code developed by W. K. Anderson (NASA LaRC) for steady compressible and incompressible Euler and Navier-Stokes equations (with one-equation turbulence modeling)
- Won Gordon Bell Prize in 1999
- Uses MPI for parallelism
- Application contains \textit{ZERO} lines of MPI
  - All MPI within the PETSc library
Why Was MPI Successful?

- It address *all* of the following issues:
  - Portability
  - Performance
  - Simplicity and Symmetry
  - Modularity
  - Composability
  - Completeness
Portability and Performance

• Hardware changes (and improves) frequently
  ♦ Moving from system to system often the fastest route to higher performance

• Performance must be competitive
  ♦ Pay attention to memory motion
  ♦ Leave freedom for implementers
    • Standard document requires careful reading
    • Not all implementations are perfect
      ‣ (When you see MPI pingpong asymptotic bandwidths that are much below the expected performance, it is the implementation that is broken, not MPI)
Simplicity and Symmetry

- Complexity should be measured in the number of *concepts*, not functions or size of the manual
- Exceptions are hard on *users*
  - But easy on implementers — less to implement
- MPI organized around a few powerful concepts
Modularity and Composability

- Modern algorithms are hierarchical
  - Do not assume that all operations involve all or only one process
  - Provide tools that don’t limit the user
- Modern software is built from components
  - MPI designed to support libraries
- Environments are built from components
  - Compilers, libraries, runtime systems
  - MPI *designed* to “play well with others”
Completeness

- Make sure that the functionality is there when the user needs it
  - Don’t force the user to start over when a new feature is needed
An Open and Balanced Process

- Balanced representation from
  - Users
    - What users want and need
      ‣ Including correctness
  - Implementers (Vendors)
    - What can be provided
      ‣ Many MPI features determined by implementation needs
  - Researchers
    - Directions and Futures
      ‣ MPI planned for interoperation with OpenMP before OpenMP conceived
      ‣ Support for libraries strongly influenced by research
Is HPC Software so Bad?

Reasoning from make:

... Finished prerequisites of target file `../../../lib/libmpich.a'.
Prerequisite `errutil.o' is newer than target `../../../lib/libmpich.a'.
No need to remake target `../../../lib/libmpich.a'.
Finished prerequisites of target file `all'.

(Note: this problem is not caused by jittering time stamps. It is simply incorrect code.)
HPC Software Issues

• Many are the same as for non-HPC software
  ♦ Performance is an additional complication

• Solutions must address the software engineering issues
  ♦ Better coding practices
  ♦ Better design (make it harder for the programmer to make mistakes)
  ♦ Encourage well-designed composition of solutions
  ♦ Balance the needs and wishes of users and implementers