Programming Environments With and Beyond MPI

James B. White III (Trey)
trey@ornl.gov

Research sponsored by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

Disclaimer

- The following opinions are not necessarily shared by ORNL, UT-Battelle, or DOE.
- Probably for good reason.
- I may not agree with them myself.

Outline

- Getting to the next prefix (peta)
- Applications at future prefixes
- Programming environments for future prefixes

Outline

- Getting to the next prefix (peta)
- Applications at future prefixes
- Programming environments for future prefixes

OpenMP is harder than MPI

- Implicit parallelism
- Private versus shared variables are a pain
- Procedures and library calls within OpenMP regions are confusing
- Easy to make quiet errors
- Easy to create obscure race conditions
- Performance is not under your control
- You already have MPI

Add OpenMP to your MPI code

- Aggregate messages for efficient use of network
- Avoid network contention
- Share cache better
- Parallelize other dimensions
- Scale to more cores
- Portable performance
  - Easy to tune thread/task ratio for each architecture

PGAS languages are harder than MPI

- Must separately synchronize by hand
- Easier to screw up synchronization
- Easier to create obscure race conditions
- Weak-to-no collectives

MPI is faster than PGAS

- For real apps on most networks
- MPI encourages packing and aggregation of messages
- MPI collectives enable optimization
- PGAS collectives are weak
  - Easy to hand-code inefficient collectives

PGAS is faster than MPI

- For global load/store networks
  - T3E, X1(E), future XT? Anybody else? Pretty please?
- Compiler understands types and array layouts
- Compiler can prefetch and schedule
- Fine-grained overlap of computation with remote-access latency
- Works wonders with vectorization
  - Tune MPI apps with incremental CAF on X1(E)

Outline

- Getting to the next prefix (peta)
- Applications at future prefixes
- Programming environments for future prefixes

Bill Camp was right about climate

- They will hit a time-step barrier
- So will others with time-dependent problems
- We must work to make Bill wrong

Implications of time barrier

- Multi-level methods (multigrid)
  - Accelerate global convergence
- Adaptive meshes, maybe
  - If your problem has *highly* localized detail
- Issues
  - Tree algorithms
  - Load balancing

Outline

- Getting to the next prefix (peta)
- Applications at future prefixes
- Programming environments for future prefixes

Hierarchies

- Deeper memory hierarchies
- Deeper parallel hierarchies in hardware
- Deeper parallel hierarchies in applications
- Need *flexible* way of mapping software hierarchies to hardware
- *Not* a new API for each new level (like OpenMP+MPI)

Baby steps

- Multidimensional co-arrays
  - Analog of Fortran array layout in memory
  - Elements differing in first dimension are nearby
  - Higher dimensions are farther away
- MPI communicators
  - User requests an array of communicators
  - User picks the array size
  - First element is communicator of nearby tasks
  - Later elements include farther tasks

Cliff diving

- Source code
  - *Specify* computation
  - Emphasize correctness
  - Specify all real dependencies, no false
- Meta-source
  - Mapping to specific architecture, performance tuning
  - Build configuration, auto-tuning configuration
  - Job configuration: I/O, diagnostics, timers
  - Fault-tolerance config (checkpoint, redundancy)
  - Hippocratic Oath: Don’t hurt correctness

Programming Environments With and Beyond MPI

James B. White III (Trey)
trey@ornl.gov

Research sponsored by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.