SAND2015-2169C

Co-design with proxy applications at the DOE NNSA Trilabs: performance of SNL proxies on modern and some future architectures

SIAM CSE

Salt Lake City

March 18, 2015

Paul Lin and Richard Barrett
Sandia National Labs

U.S. DEPARTMENT OF ENERGY
NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Historical perspective

- Traditional relationship between hardware vendors and application "app" developers: machine shows up and app developers forced to get their code to run well on it
  - but the gap between peak performance and actual app performance keeps widening (e.g. FEM unstructured mesh app sparse iterative solver and multigrid preconditioner achieves ~1% of peak)
- Future architectures so radically different that in order to get reasonable performance, all teams need to work together: hardware, runtime environment, programming models, compilers, application developers → "co-design"
  - Others have different definitions for "co-design"
  - For this talk, we will use the above ASC view of co-design
  - App developers would prefer it if other teams just took their multi-million line codes and got them to work well; not a realistic expectation

Performance proxies for applications

- If a performance proxy that captures certain key performance characteristics of the app can be developed, interactions between the apps teams and other teams could be greatly facilitated
- Need a proxy that can represent key performance aspects
  - Careful design needed
  - Even properly designed proxy can only be used within scope of design; otherwise results will be misleading
  - Goal is to provide insight
  - Provides concrete software for hardware vendors and apps developers to communicate
  - Never forget that the app proxy is not the app
- Assume proxy has been properly designed and "validated"
  - E.g. see "Assessing the Validity of the Role of Mini-Applications in Predicting Key Performance Characteristics of Scientific and Engineering Applications," R. Barrett et al. *Journal of Parallel and Distributed Computing*, Vol 75, Jan 2015, pp 107-122

Focus of talk

- Effort during FY14 (DOE ASC "milestone"): explored how some tri-labs proxies perform on modern and future architectures
  - Tri-labs: Los Alamos (LANL), Lawrence Livermore (LLNL), Sandia (SNL)
  - Goal: provide some insight what to potentially expect for the future
- Effort focused on two proxies per lab
  - LANL
    - SNAP: Deterministic Sn Transport
    - PENNANT: Unstructured hydrodynamics
  - LLNL
    - UMT: Deterministic Sn transport
    - MCB: Monte Carlo particle transport
  - SNL (focus of this talk)
    - MiniFE: Implicit unstructured finite elements
    - MiniAero: explicit high Mach aerodynamics

Contributors (SNL work)

- Richard Barrett, Carter Edwards, Ken Franko, Si Hammond, Glen Hansen, L., Mahesh Rajan, Dylan Stark, Christian Trott, Courtenay Vaughan, Patrick Xavier, Alan Williams, and others.

- Several vendor staff, including Mike Davis (Cray), Duncan Roweth (Cray), Justin Lutjiens (Nvidia), Intel Phi team, and others.

- Local testbed support, including Jim Laros, Sue Kelly, system admin teams, and others.

- Thanks to LC Sequoia BG/Q team for support

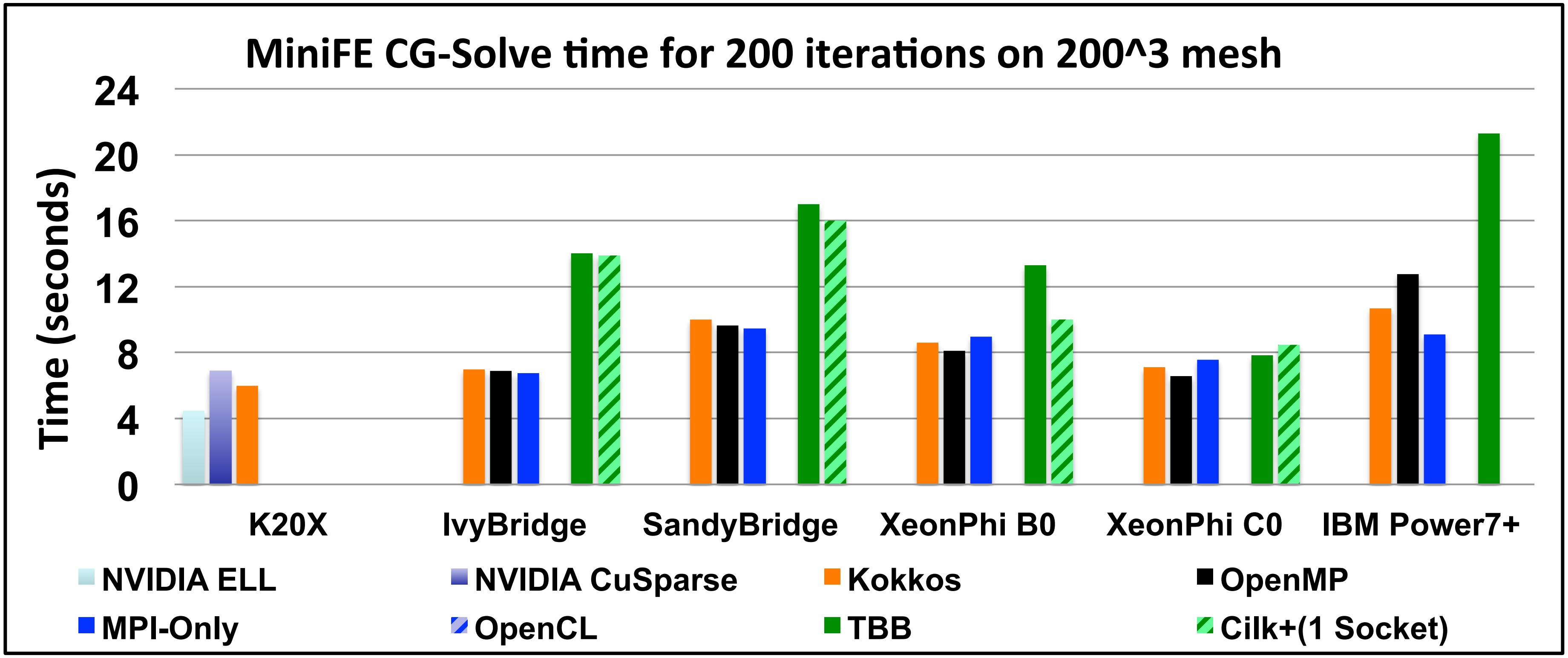
Mantevo project (mantevo.org)

miniapp or miniDriver		
CleverLeaf (AWE)	Eulerian on structured grid with AMR	
CloverLeaf, CloverLeaf3D (AWE)	Compressible Euler eqns, explicit 2nd order accurate	
CoMD (LANL/LLNL)	Molecular dynamics (SPaSM)	
EpetraBenchmarkTest	Exercises Epetra sparse and dense kernels.	*3.0 release*
HPCCG	Unstructured implicit finite element	
miniAero	3D unstructured FV R-K 4th order time, inviscid Roe Flux	
miniAMR	Adaptive mesh refinement of an Eulerian mesh	
miniFE	Implicit finite element solver	
miniGhost	FDM/FVM explicit (halo exchange focus)	
miniMD	Molecular dynamics (Lennard-Jones)	
miniSMAC2D	FD 2D incompressible N/S on a structured grid.	
miniXyce	SPICE-style circuit simulator	
PathFinder	Signature search	
TeaLeaf (AWE)	Heat conduction with implicit solvers (CG and Cheby) on a 5-pt stencil.	
miniExDyn-FE	Explicit Dynamics (Kokkos-based)	
miniITC-FE	Implicit Thermal Conduction (Kokkos-based)	
phdMesh	Explicit FEM: contact detection	

Implicit finite element proxy: miniFE

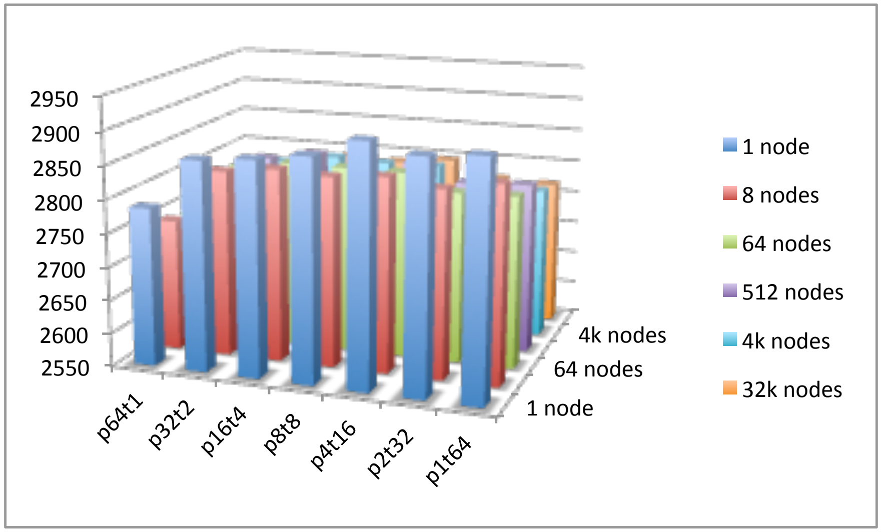
- SNL has many implicit FE apps
- Steady-state 3D heat equation (Poisson equation) in cube
- Structured mesh, but data stored as unstructured mesh
  - had to tell one vendor that they were not allowed take advantage of the underlying structured mesh
- Finite element method with hexahedral elements
- FEM matrix and RHS assembly
  - too simple for real apps; more realistic assembly needed
- Symmetric matrix solved by CG (no preconditioner)
  - Lack of preconditioner: big weakness
  - No multilevel/multigrid---critical for scaling
- Implemented in ~20 variants in 10 programming mechanisms
- Variants for several hardware platforms including vendor simulators

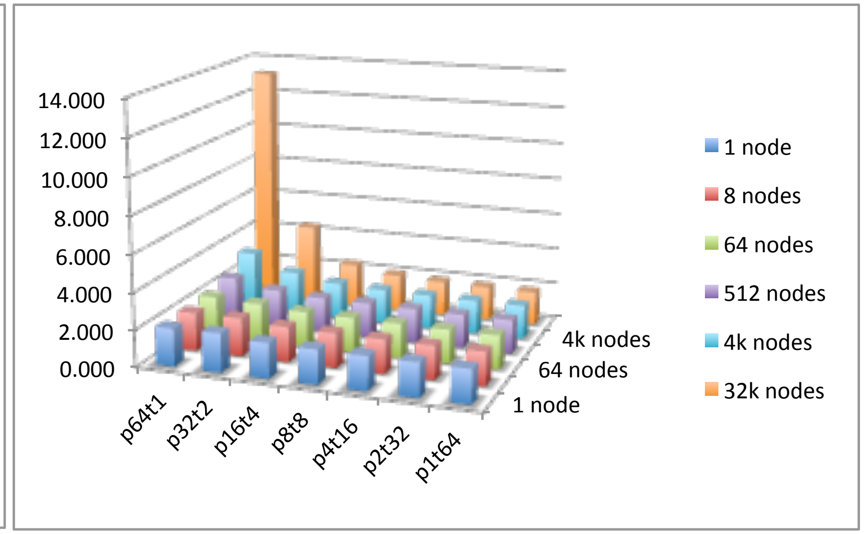
miniFE single node performance



- Kokkos provides performance portabillity
  - Same code implemented using Kokkos can be run on a CPU, GPU or Xeon Phi
  - Don't give up much performance (10-20%) vs. writing own code or vendor library

miniFE weak scaling to 32k nodes BG/Q





Average Mflops/node

- Sparse iterative solve bandwidth bound, so not much variation

Average memory/node (GB)

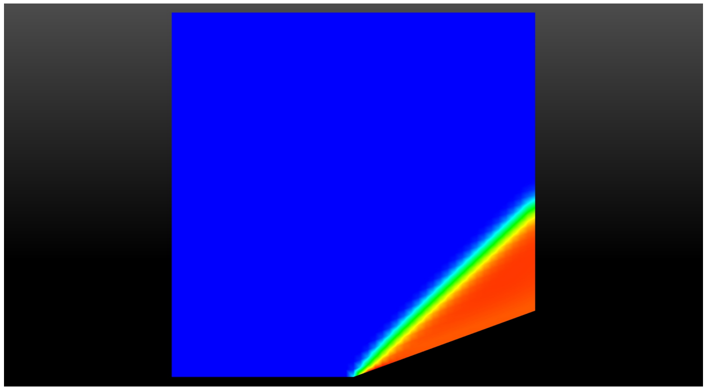
- 1 MPI task/node (64 threads): slow memory growth as scale
- 64 MPI tasks/node: memory rapidly growing; 32k nodes (2 million MPI tasks) close to maxing out memory
- MPI+OpenMP threads can significantly save memory

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration

Sandia National Laboratories

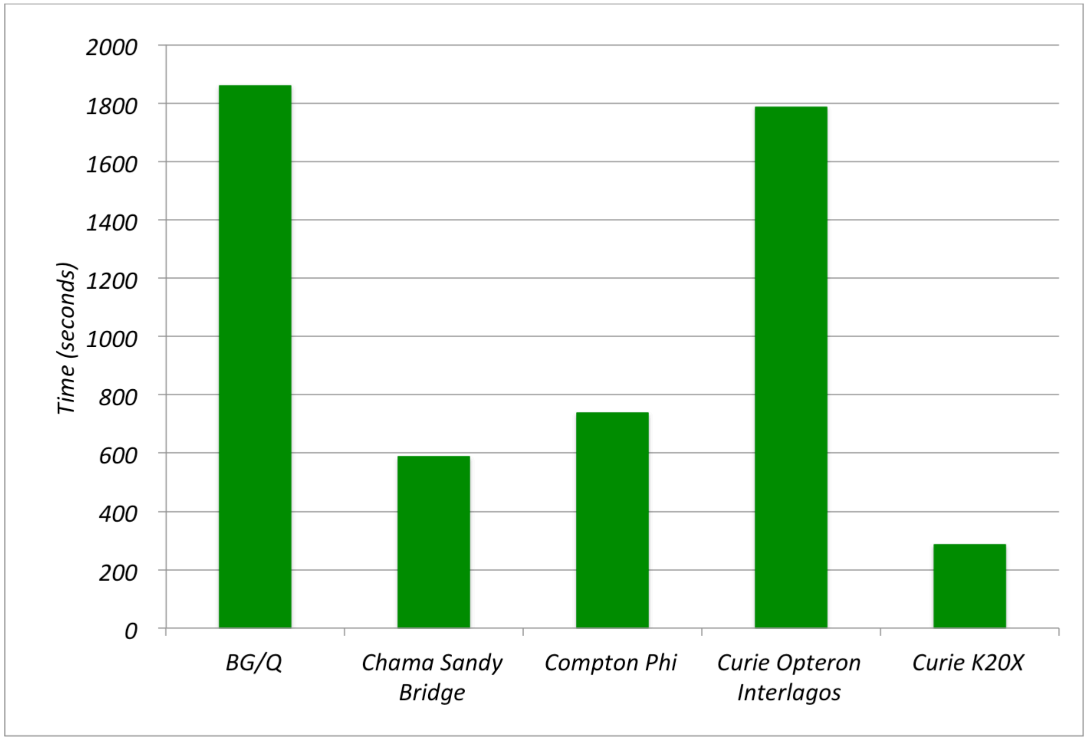
Compressible flow app proxy: miniAero

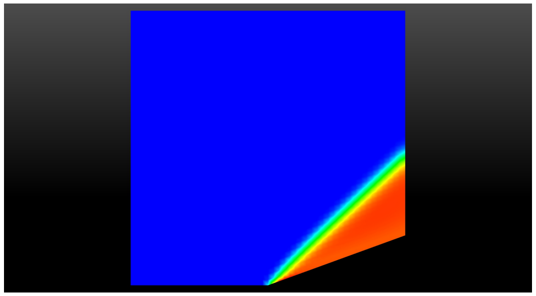
- 3D unstructured finite volume
- Runge-Kutta 4th order time
- Inviscid Roe Flux



- Based on Kokkos
- Physics kernels are functors -> flexible
- Use of templates for device and algorithm choices
- Still under development
  - Additional physics (LES, etc.)
  - Point implicit solver

miniAero single node performance





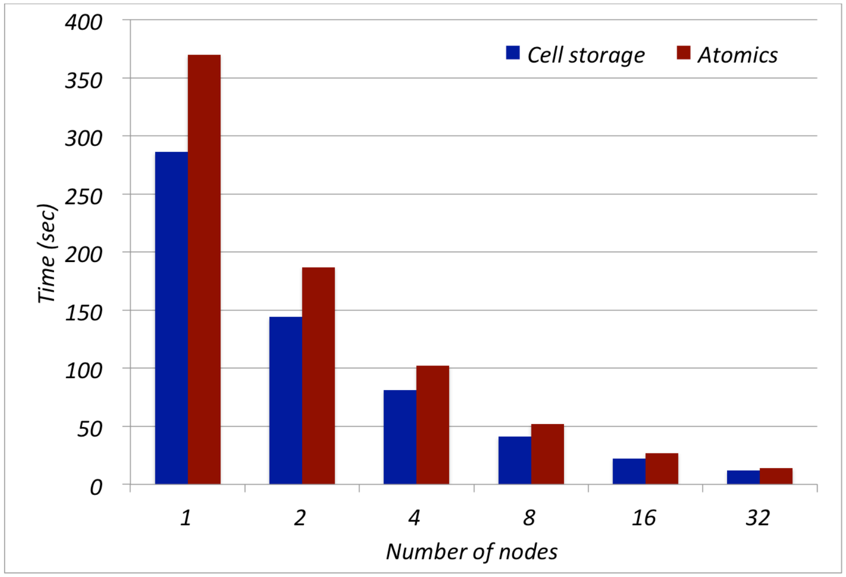
- ~4.2M elements
- 500 time steps

Platform	Processor	Clock Speed (GHz)	Runtime (sec)
BG/Q	16 core/64 threads	1.6	1861
Chama	Intel Xeon Sandy Bridge, 2x8 cores	2.6	587
Compton	Intel Xeon Phi 7120 (KNC), 57 cores	1.1	738
Curie Opteron	16 cores AMD Interlagos, 2x8 cores	2.1	1787
Curie	Nvidia K20X Kepler, 2688 cores	0.7	286

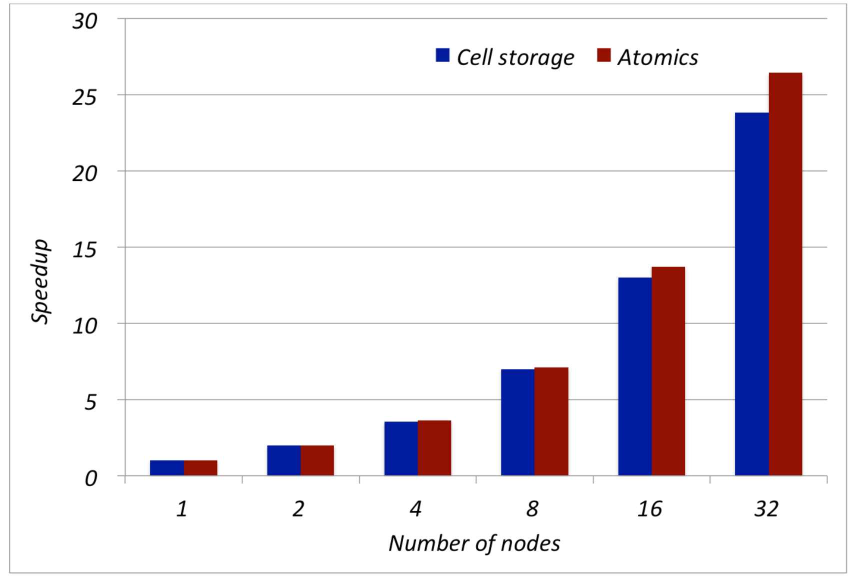
U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration

Sandia National Laboratories

miniAero multiple nodes MPI+GPU



Time (sec)



Speedup

Summary

- GPU speedups can be significant
- FV (explicit) amenable to threading, both CPU and GPU
  – Thread safety required
- Further performance evaluation and tuning needed
- Additional hardware testing needed (Phi, BG/Q, Titan)
- Kokkos – promising for heterogeneous architectures

Summary

- As part of work performed to fulfill an NNSA ASC milestone, the trilabs performed studies of proxy apps that concerned performance on current and future platforms
- Talk focused on two SNL proxies: miniFE and miniAero
  - Proxies representative of important SNL apps
  - Demonstrated that Kokkos provides performance portability
    - Typical “rule of thumb” is that app developers can get 80-90% of performance of native choice, but get this across multiple hardware choices
  - MPI + threads effective
    - But hard to get performance win over MPI-only for CPUs
- Proxy apps have demonstrated value, but full app work critical to understand full complexity
  - Never forget that the proxy app is not the app

Future work

Lots and lots of it; a small sampling…

- More proxy apps
  - E.g. “miniFEassembly” as miniFE matrix assembly is not representative of app
- Ensure proxy apps are really representative of app
- More detailed and extensive studies of proxy apps on architectures
- Studies on additional architectures
- Task-based parallelism approaches
  - Unitah, Legion, Charm++, etc.
  - Co-design issues, especially with respect to runtime systems

Proxy applications value

- What is the value of proxy applications?
  - effective means to isolate specific issues for current and future systems
  - Greatly ease communication between computational scientists, computer scientists and computer vendors
  - Enable rapid exploration of programming models, abstraction techniques, and optimization approaches in a quasi-realistic context, for subsequent adoption by a full application
- What are their short-comings?
  - Too easy to misuse
  - Simplicity can be misleading: a single-physics proxy application may be significantly easier to optimize on challenging architectures (e.g. GPUs) than multi-physics applications with their more dynamic behavior
- What recommendations could be followed to increase their value?
  - Better documentation on how to do scaling studies (particularly weak scaling), physics (parameter ranges that, etc.)
  - Better documentation on how to vary physics (i.e. how to select parameter ranges that test the limits of interest wrt the target computational science)
  - Caveat: Can all of the above be done without a testing framework as complex to grasp and maintain as a full application?

Thanks For Your Attention!

Paul Lin (ptlin@sandia.gov)

Acknowledgment

- This work was performed under the DOE NNSA ASC milestone: *"Milestone #4875: Evaluate Application Performance on Advanced Architectures,"* programmatic leads: Richard Barrett (SNL), David Daniel (LANL), Todd Gamblin (LLNL), Mike Glass (SNL), Rob Hoekstra (SNL), Louis Howell (LLNL), Christoph Junghans (LANL) Al McPherson (LANL), and Rob Neely (LLNL)

The authors gratefully acknowledge funding from the DOE NNSA Advanced Simulation & Computing (ASC) program