SAND2012-4346P

Exceptional service in the national interest

Sandia National Laboratories

Characterize the Role of the Mini-Applications in Predicting Key Performance Characteristics of Real Applications ASC L2 Milestone 4465 Panel Review

D.W. Doerfler (lead), R.F. Barrett, P.S. Crozier, M.A. Heroux, P.T. Lin, H.K. Thornquist, T.G. Trucano, C.T. Vaughan

May 23, 2012
SAND 2012-TBD

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Text of Milestone

Description: The Mantevo project includes a set of application proxies, referred to as "mini-apps," and designed by code developers to represent key runtime performance characteristics of their applications. SNL will analyze two of these mini-apps to determine how well they represent the full application programs. Specifically, SNL will profile the runtime performance of the mini-app and application, characterizing the relationship between the two on at least two HPC platforms (including Cielo).

Contribution to Stockpile Stewardship: The long term usefulness of ASC codes to the Stockpile Stewardship program will necessitate that the codes evolve as supercomputer architectures change. Dramatic changes to full production codes are expensive and high risk. Applications proxies can play an important role in determining the evolutionary path of production codes without incurring the high overhead of working with the full code.

Review committee

- James A. Ang, Scalable Architectures (1422) manager
- Teddy D. Blacker, Simulation Modeling Sciences (1543) manager
- Robert J. Hoekstra (lead), Scalable Algorithms (1424) manager
- Allen L. McPherson, Computer Scientist, Los Alamos National Laboratory
- William J. Rider, Computational Physicist, Computational Shock and Multiphysics (1443)
- Charles (Bert) Still, Computational Physicist, Lawrence Livermore National Laboratory

Outline

- Executive Summary
- Motivation
- Methodology
- Dives on the data
- Summary and future plans

Executive Summary

- We have defined a methodology for comparing apps and miniapps, providing a framework for reasoning about important performance-impacting issues.
  - These issues have been identified through performance analysis, understanding of applications and the machines on which they are designed to run.
  - An ongoing process.
- We have applied the methodology to four applications and their representative miniapp on Cielo and at least one other platform.
- Using miniapp reference implementations (mpi-everywhere), we have made observations regarding where a miniapp is and is not representative of key metrics in its associated application.

Executive Summary cont'd

We find that, among other things, for

- LAMMPS: miniMD
  - is a strong proxy for the three main phases of the Lennard-Jones atomic interaction: force, neighbor binning (under certain conditions), communication, but
  - is not broadly representative of molecular dynamics.
- Charon: miniFE
  - is a strong proxy for node memory behavior, but
  - is not designed to capture multi-level preconditioning.
- CTH: miniGhost
  - represents inter-process boundary exchange, but
  - is not intended to represent computation, esp. AMR.
- Xyce: miniXyce
  - is a compact application, but
  - Is not a miniapp.

Based on the above conclusions and the analysis to follow we believe we have satisfied the milestone requirements.

Motivation

- Applications and the computers they must run on are complex, including
  - limited access, shared resources, dynamic, affected by measurement intrusion and other non-deterministic effects.
- Miniapps provide a context for reasoning about key application performance issues.
- It’s a journey, not a destination.

Mantevo project

- Outgrowth of questions regarding Trilinos solvers project.
- Provides application-relevant contexts for tractably exploring new computing environments, throughout the codesign space.
- A miniapp is order 1k SLOC proxy for a key application performance issue, developed and owned by application team, open source, designed to be modified.

Under what conditions does a miniapp represent a key performance characteristic in a full app?

We have developed a methodology that adheres to the spirit of experimental validation:

- App: “real world”, Miniapp: “model”.

Requires

- extensive knowledge of, and experience using, developing, executing, instrumenting, analyzing, measuring, maintaining, and extending multi-scale, multi-physics scientific and engineering application software, targeting highest performance computing platforms, and
- a strong understanding of the miniapps and their intended use: what they ***are*** intended to represent and what they ***are not*** intended to represent, and

Provides a formal validation methodology that lets us examine experimental and predicted data.

Toward building a body of evidence

Where we…

- …were: “Trust me.”
- …are: “Middle of the beginning.”
- …going: “Continue building a body of evidence.”

Computing Environments

- Cielo : Cray XE6, ASC capability machine. 143,104 cores = 8944 x 2 x 8 AMD Magny-Cours@2.4 GHz + Gemini 3d torus.
  - Muzia : Cray XE6, surrogate for Cielo. (320 cores)
- Red Sky: Sun/Oracle configured capacity machine. 18,544 cores = 2318 x 2 x 4 Intel Nehalem@2.93 GHz + Mellanox IB 3d torus.
- Chama: Appro configured TLCC2. 19,712 cores = 1232 x 2 x 8 Intel Sandy Bridge@2.6 GHz + Qlogic IB fat tree.
- Workstations
  - 2 x 4 Intel Nehalem 5560@ 2.8 GHz processors.
  - 2 x 4 Intel Nehalem 5570@ 2.93GHz processors.
  - 2 x 8 AMD Magny-Cours 6136@2.4 GHz
  - 2 x 12 AMD Magny-Cours @2.1 GHz

V&V: General approach

1. define the set of tests,
2. explain why the test is important, then
3. present evaluation criteria (or approaches), and ideally explain how you determine whether you passed or failed a test.

Every case different

Methodology for Assessing the Validity of MiniApps

For each test, defined *diagnostics* $\{D\} = D_1, D_2, \ldots, D_{n,}$

baseline observations $\{B\} = B_1, B_2, \ldots, B_{m,}$ and

miniapp measurements $\{A\} = A_1, A_2, \ldots, A_n$, for $n \geq m$.

Then

$$X_i = f_i\,(B_i, A_i)\,, \text{ for all } i$$

$$V_i = \begin{cases} \textit{predictive}, \text{ for } T^1_i < X_i < T^2_i \\ \textit{caution}, \text{ for } T^2_i \leq X_i < T^3_i \\ \textit{not predictive}, \text{ for } X_i \leq T^1_i \text{ or } X_i \geq T^3_{i,} \end{cases}$$

for thresholds T.

Thresholds are defined in terms of each test and how it can be evaluated

May be based on

- quantitative metrics,
- qualitative metrics, and/or
- judgment, and
- may compel us to dig deeper, wider, etc. to improve confidence.
- For this context thresholds are discussed rather than defined.

Molecular Dynamics: LAMMPS and miniMD

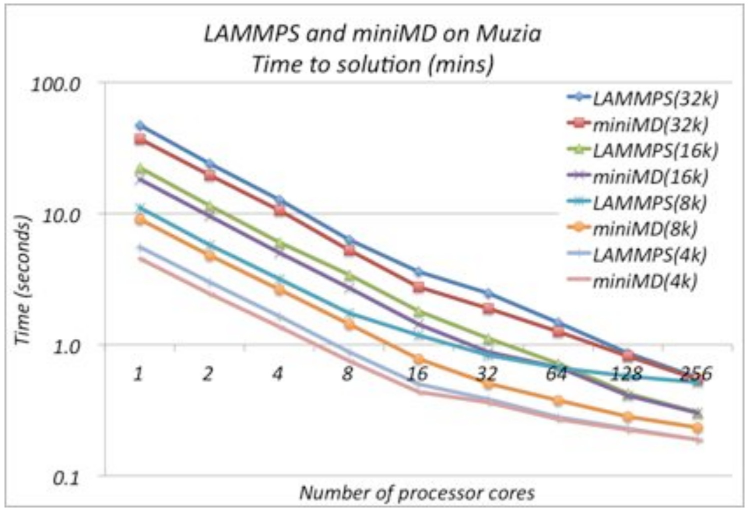
- Lennard-Jones atomic interaction

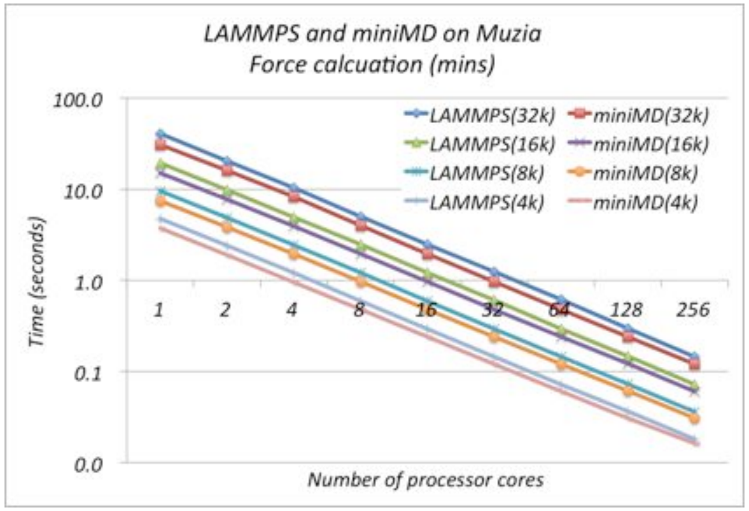
- Goals :
  - Effective processor and inter-node performance
  - Demonstration of methodology

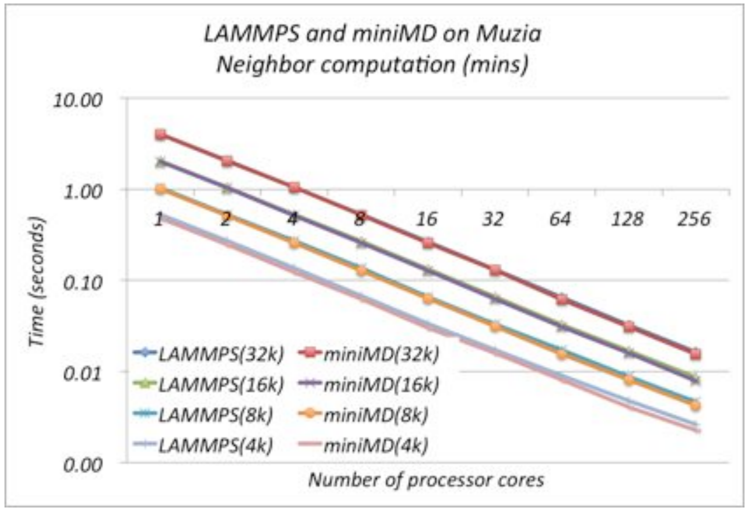
- Three main phases:
  - Force calculation
  - Neighbors
  - Inter-process communication, plus
  - Overall time to solution

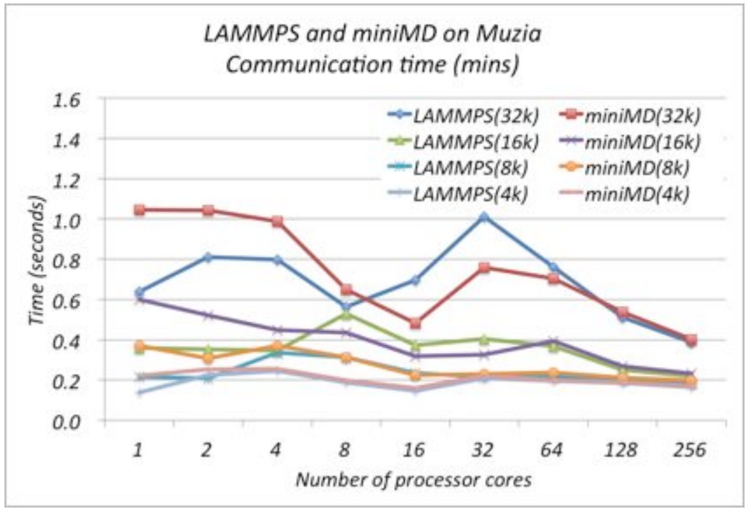
- Diagnostics: total and each phase time
- Metric: proportional difference in time

miniMD as predictor for LAMMPS Cray XE6 (Muzia)

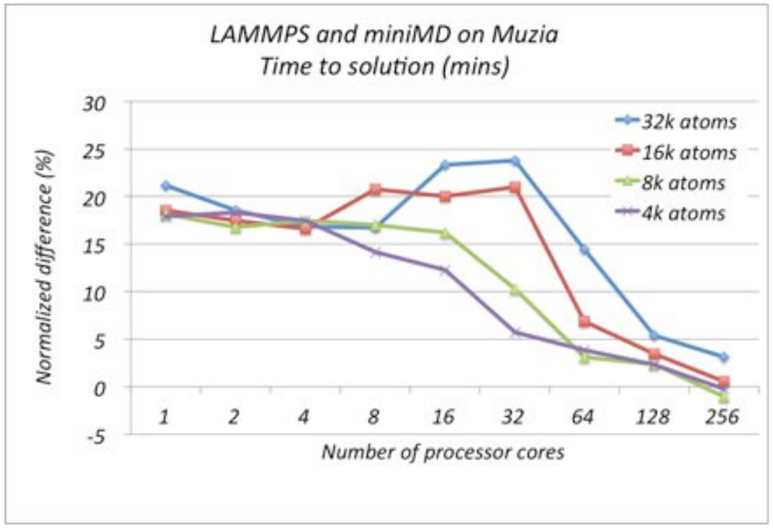


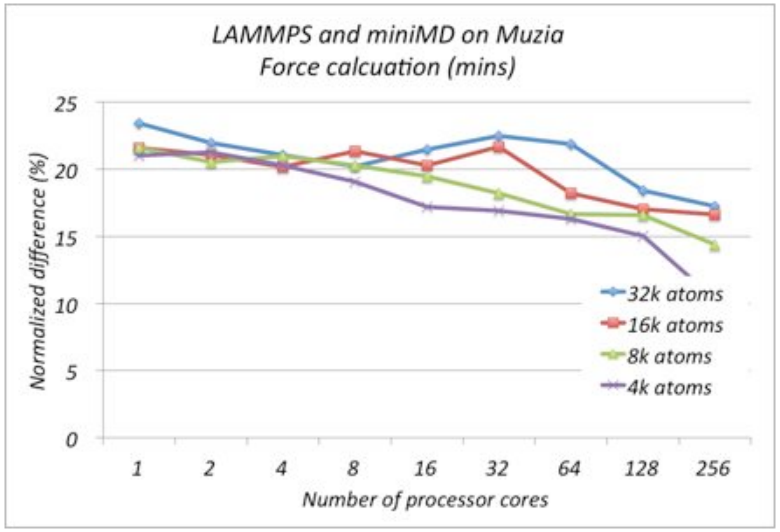


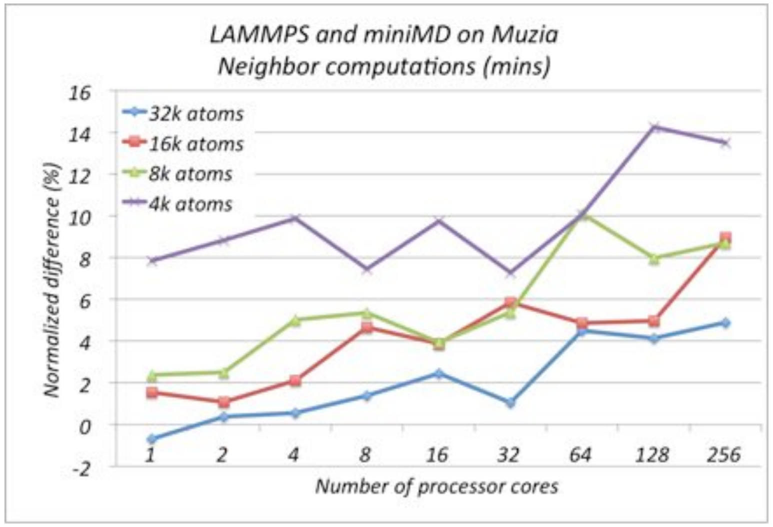


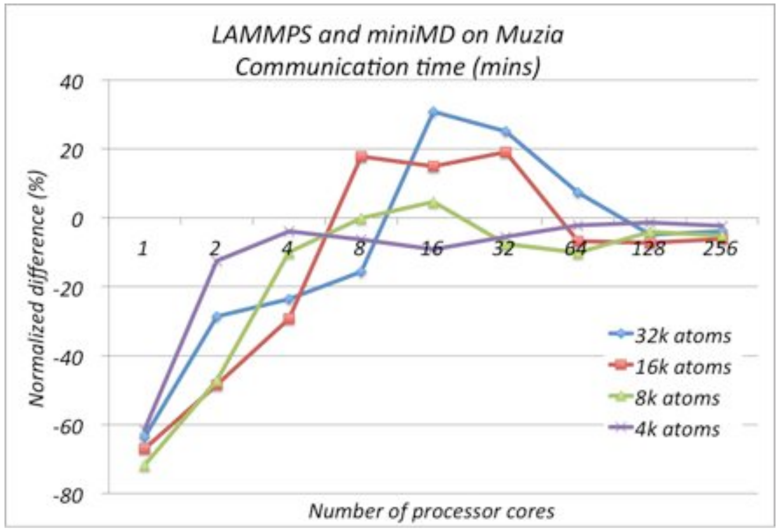


miniMD as predictor for LAMMPS Cray XE6 (Muzia) : (L-m)/L

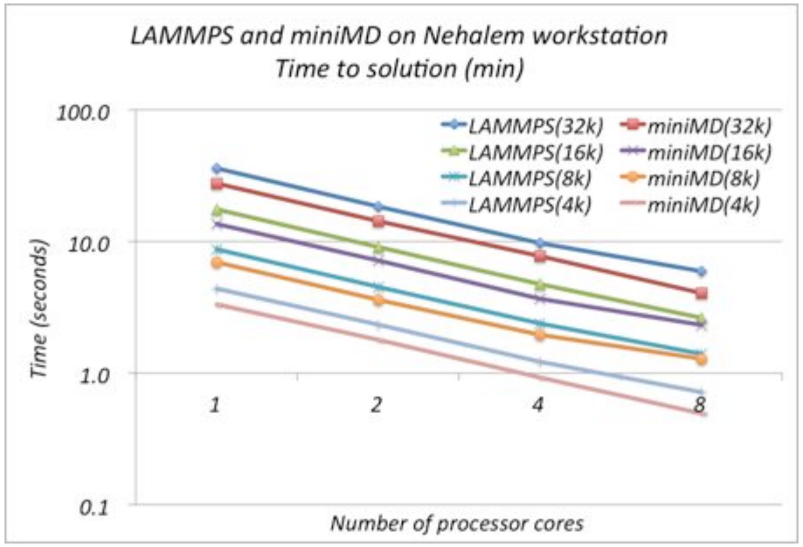


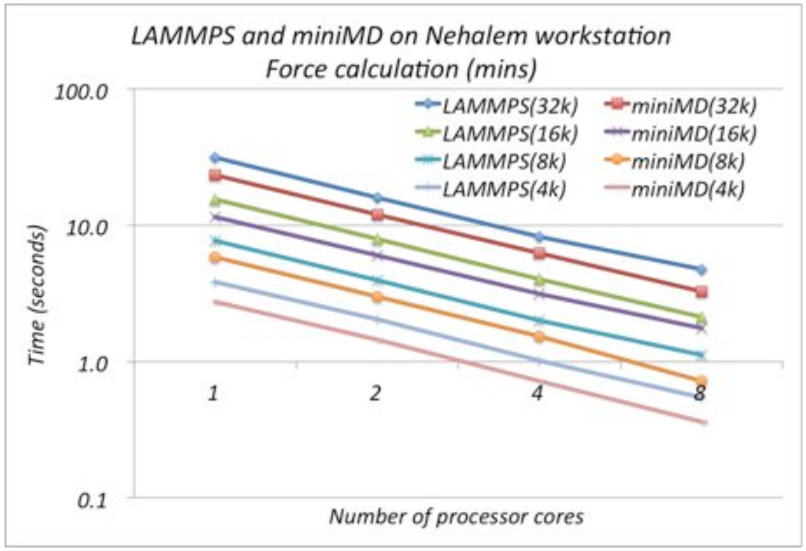


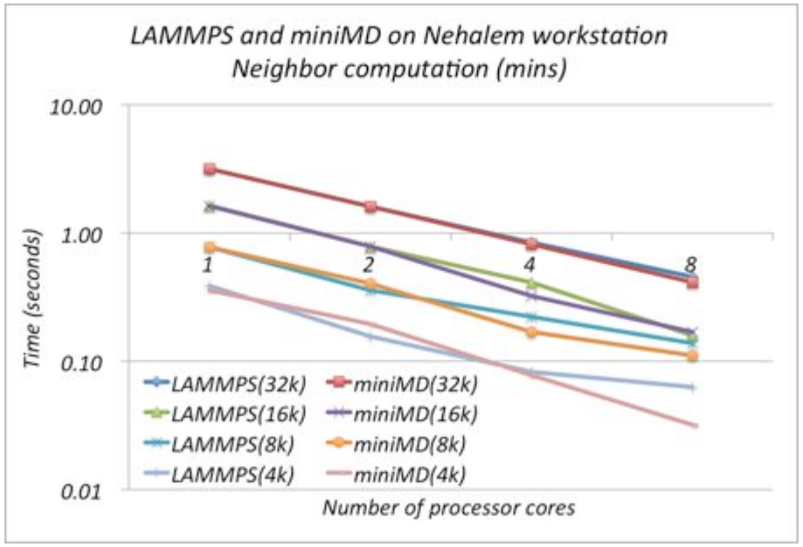


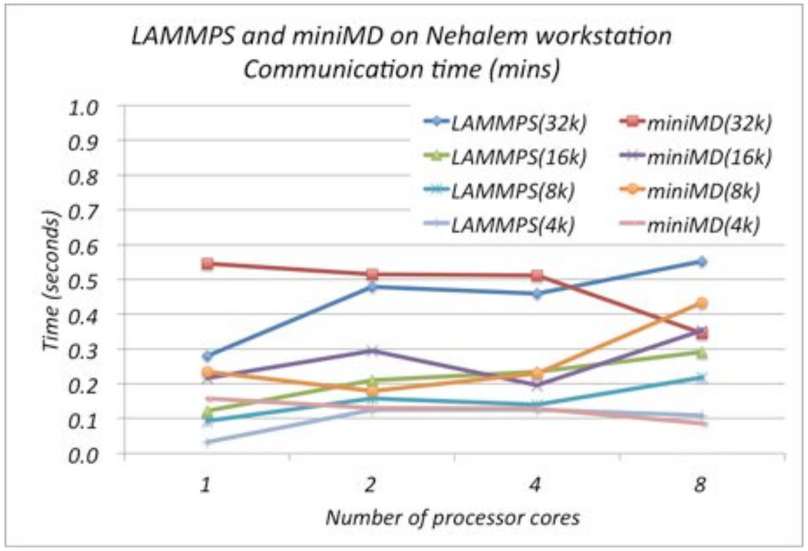


miniMD as predictor for LAMMPS Nehalem Workstation

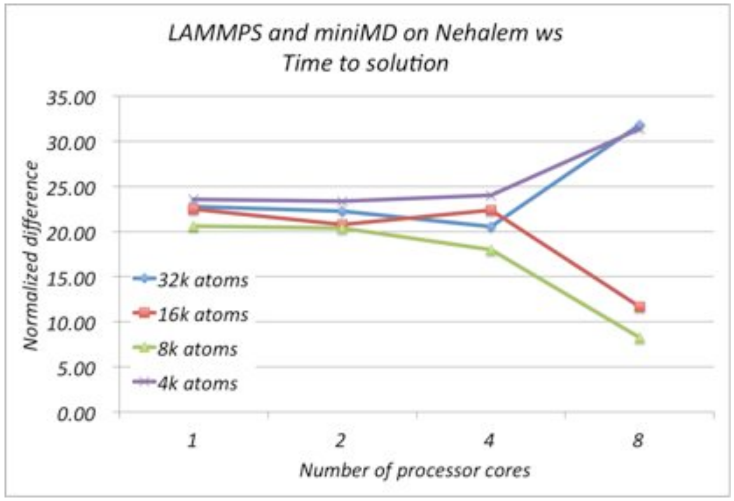


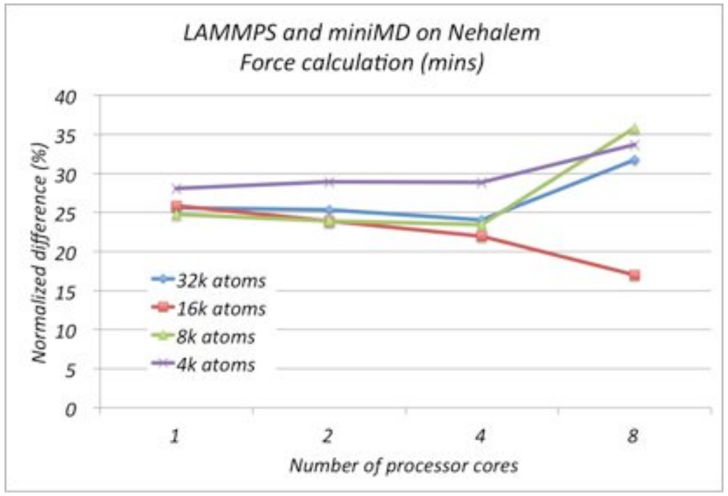


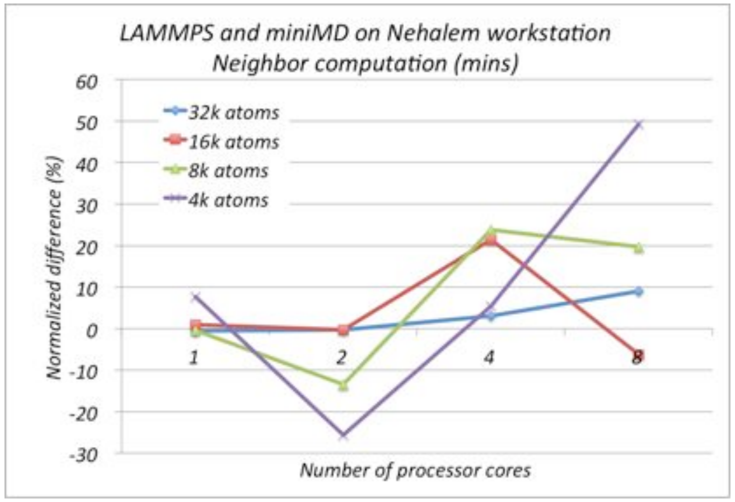


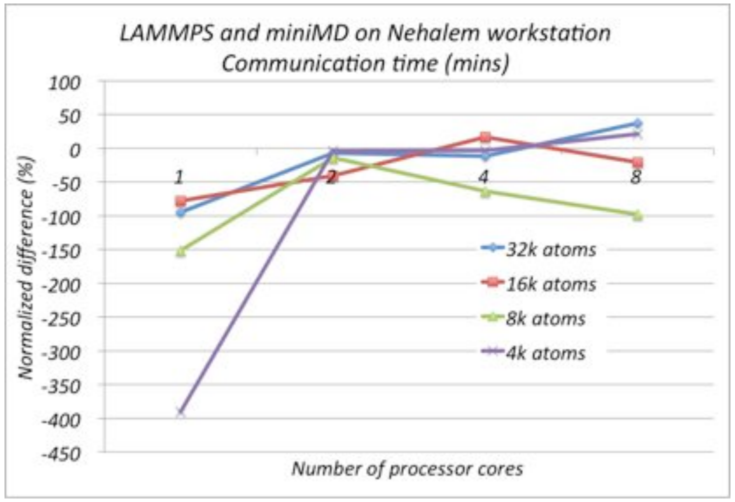


miniMD as predictor for LAMMPS Nehalem Workstation : (L-m)/L







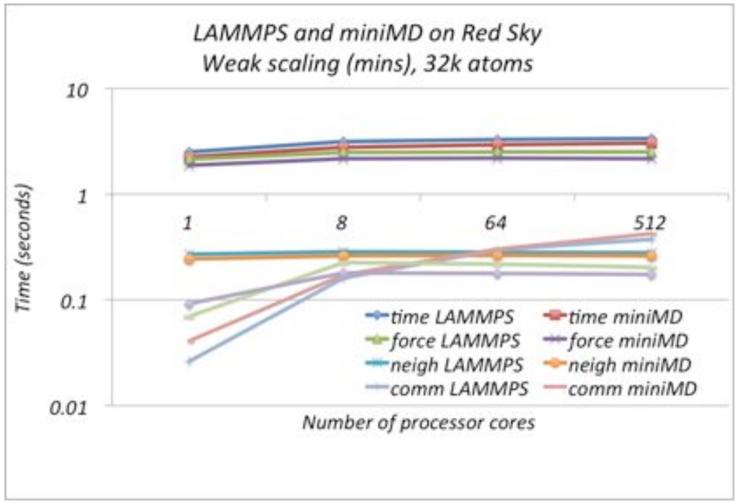


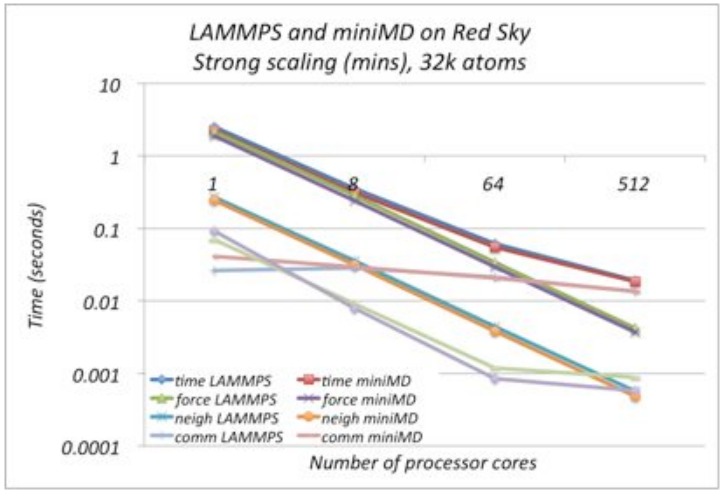
miniMD as predictor for LAMMPS Red Sky (Neh + IB)

Weak scaling

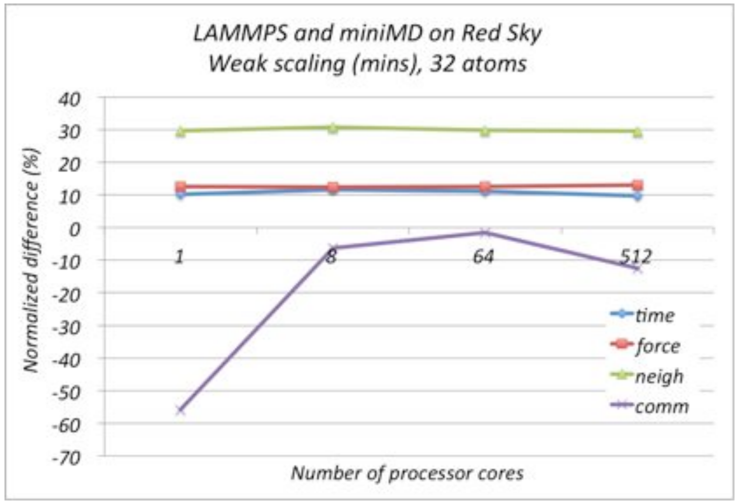
Strong scaling

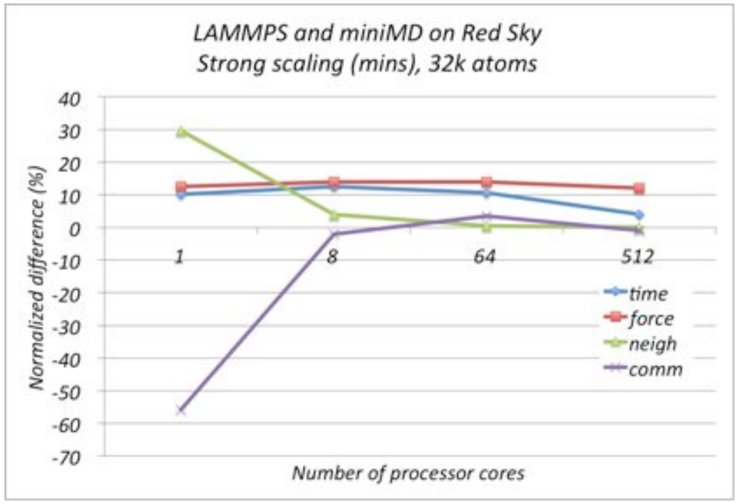
Time





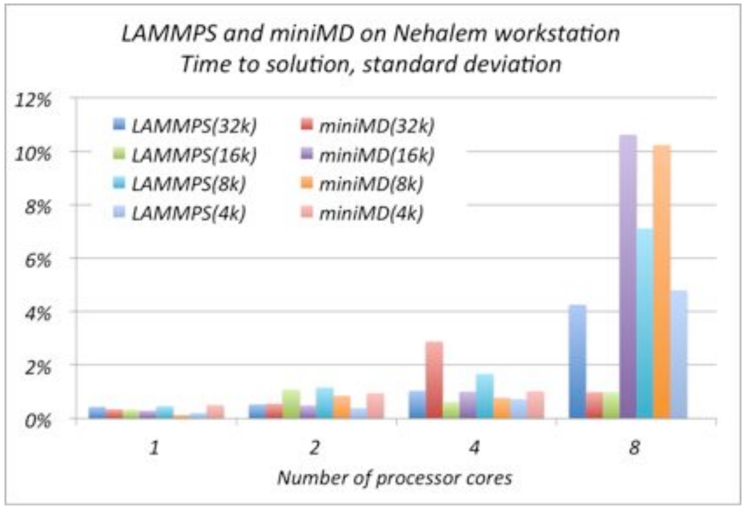
(L-m)/L

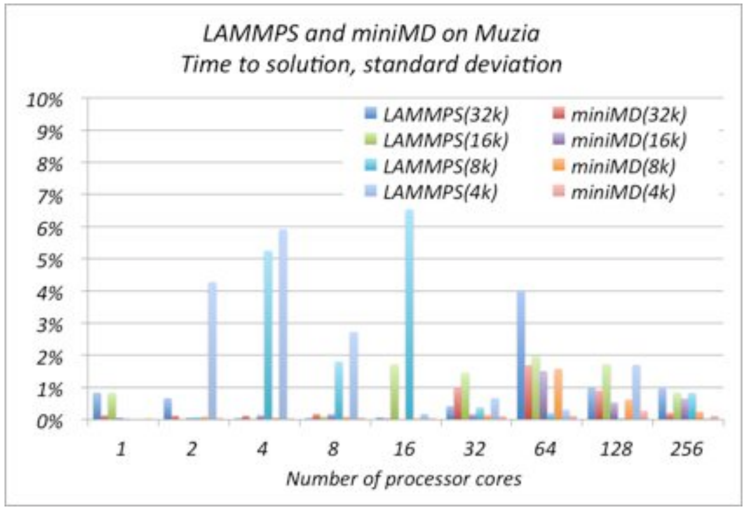


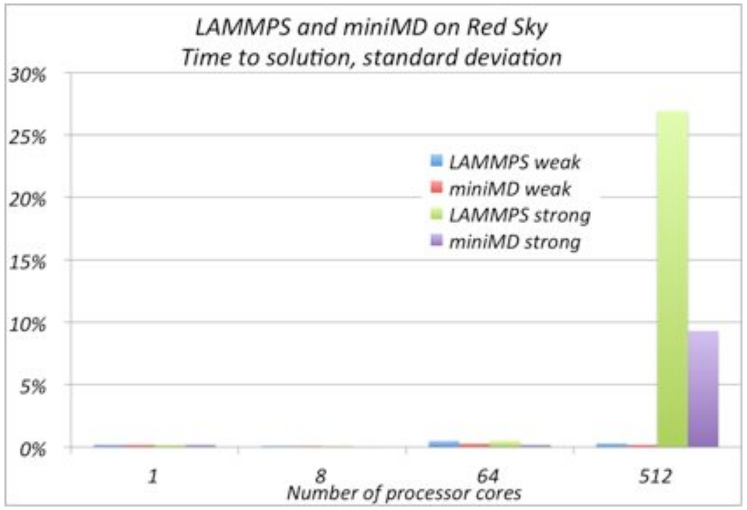


miniMD as predictor for LAMMPS

- Time to solution
- % standard deviation
- 3 trials for each







Force calculation code

```
for (i = 0; i < nlocal; i++) {
   neighs = neighbor.firstneigh[i];
   numneigh = neighbor.numneigh[i];
   xtmp = x[i][0]; ytmp = x[i][1]; ztmp = x[i][2];
   for (k = 0; k < numneigh; k++) {
     j = neighs[k];
     delx = xtmp - x[j][0]; dely = ytmp-x[j][1]; delz = ztmp-x[j][2];
     rsq = delx*delx + dely*dely + delz*delz;
     if (rsq < cutforcesq) {
       sr2 = 1.0/rsq;
       sr6 = sr2*sr2*sr2;
       force = sr6*(sr6-0.5)*sr2;
       f[i][0] += delx*force;
       f[i][1] += dely*force;
       f[i][2] += delz*force;
       f[j][0] -= delx*force;
       f[j][1] -= dely*force;
       f[j][2] -= delz*force;
}}
```

LAMMPS/miniMD Conclusions

- Data was collected on a multicore workstation, Muzia/XE6 and Red Sky.
- Metrics investigated were time to solution and timings for the three main phases: force, set neighbors and comm.

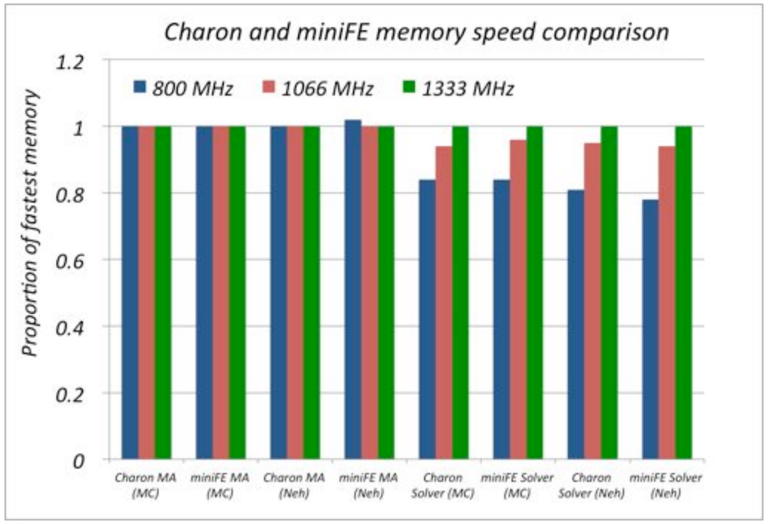
We find that

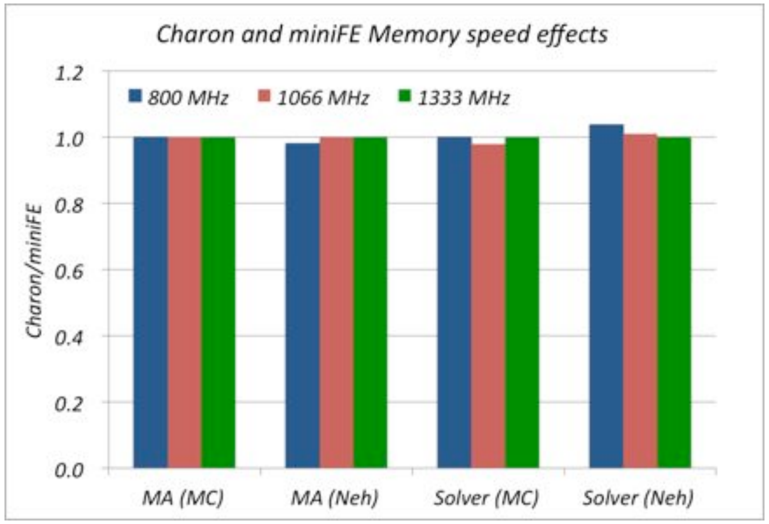
- Force is the dominant computational phase and is shown to be representative at all scales and problem sizes.
- Neighbor calculation becomes more representative as the number of atoms in the problem increases.
- Communication phase becomes more representative as the problem size increases.

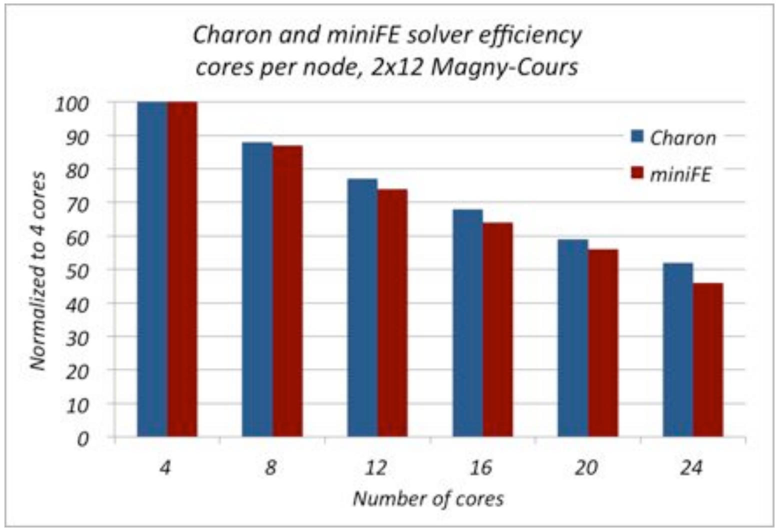
Implicit Finite Element Method Charon and miniFE

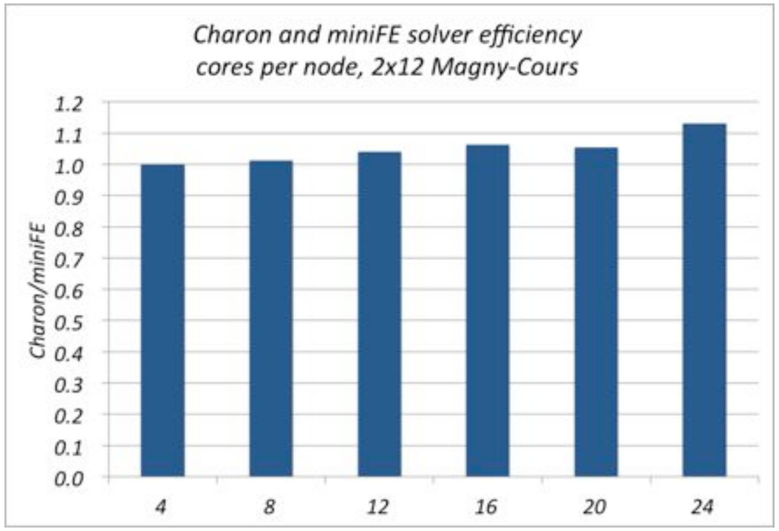
- Unstructured mesh, (Newton-)Krylov dominates.
- Goals:
  - Improved understanding of node performance.
  - Improved performance at higher scales
- Two main phases:
  - Matrix assembly
  - Solve linear system
- Diagnostics:
  - Memory speeds
  - Cache performance
  - Scaling

Charon and miniFE Diagnostic: memory speeds

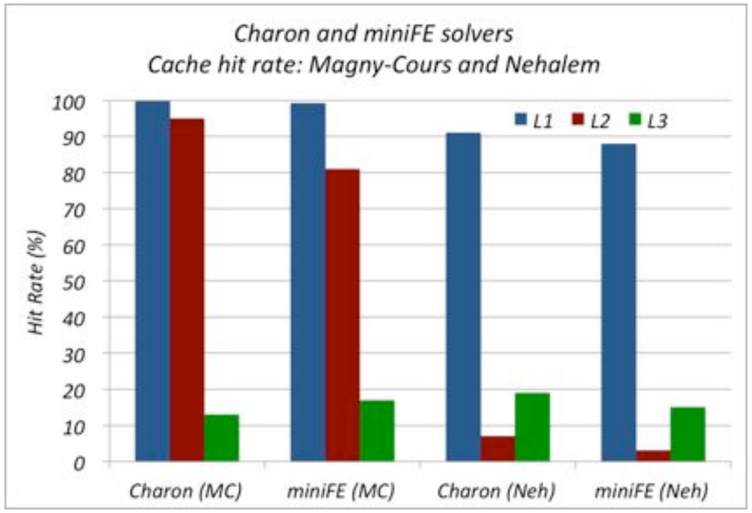


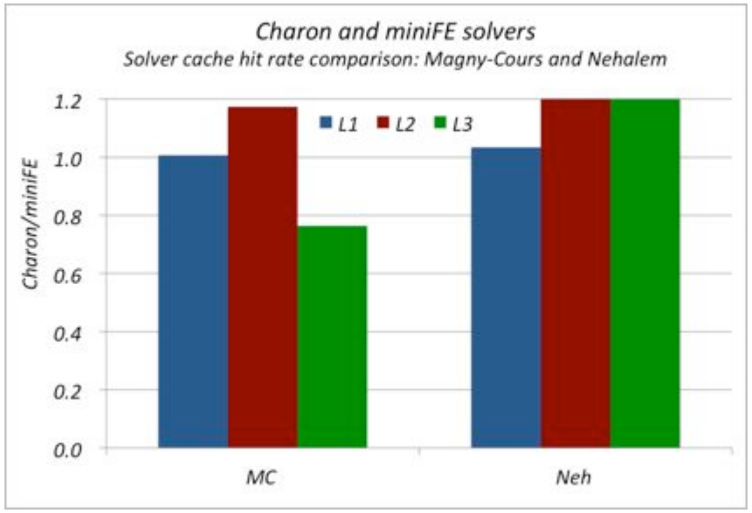


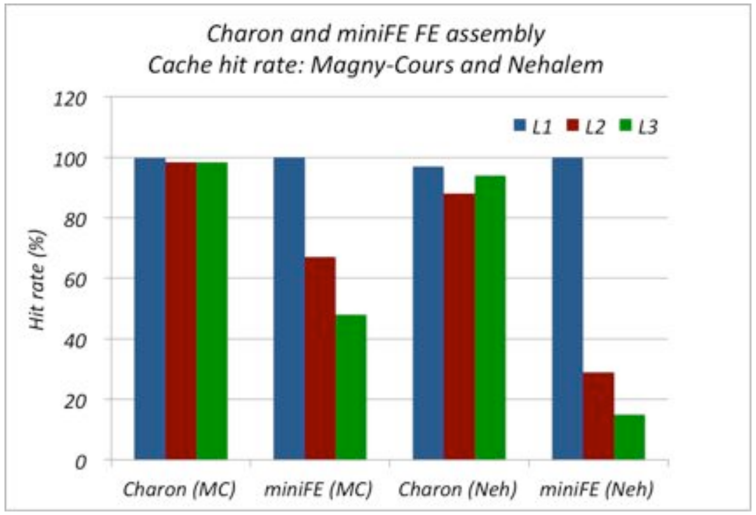


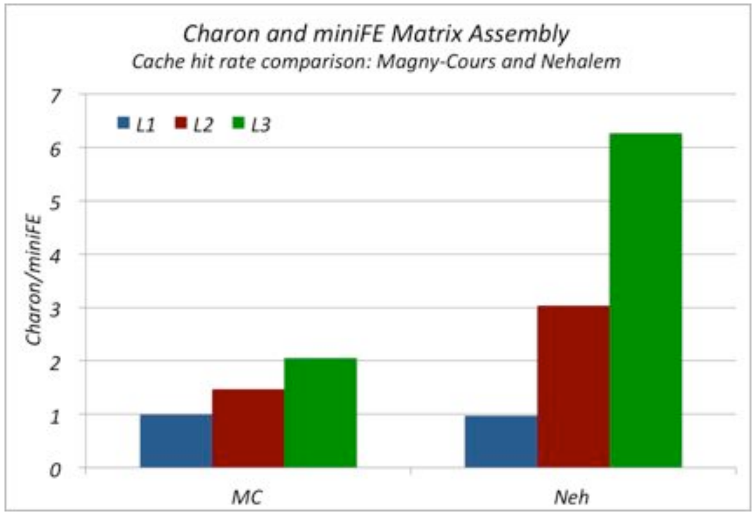


Charon and miniFE
Diagnostic: cache performance



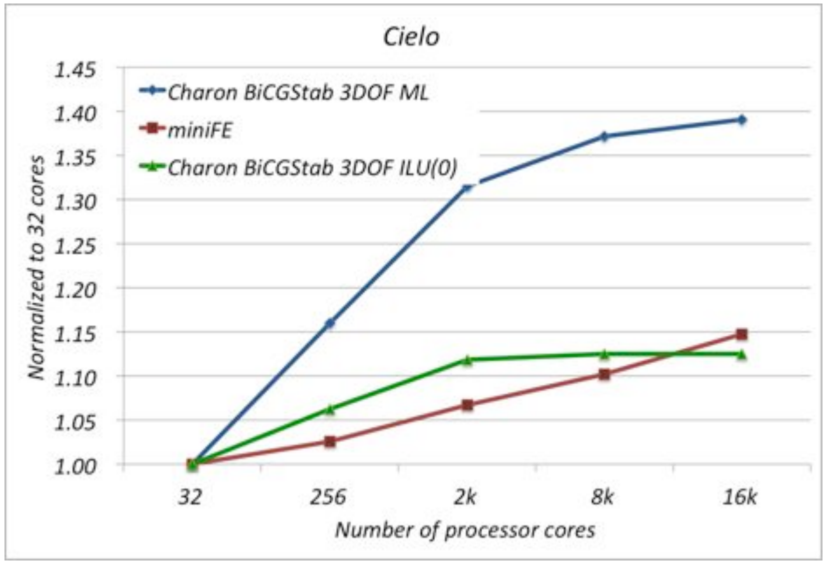


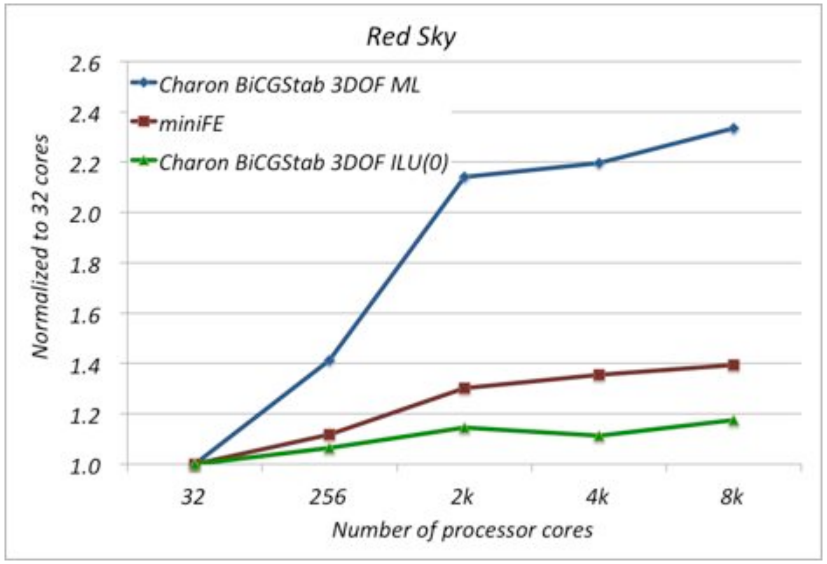


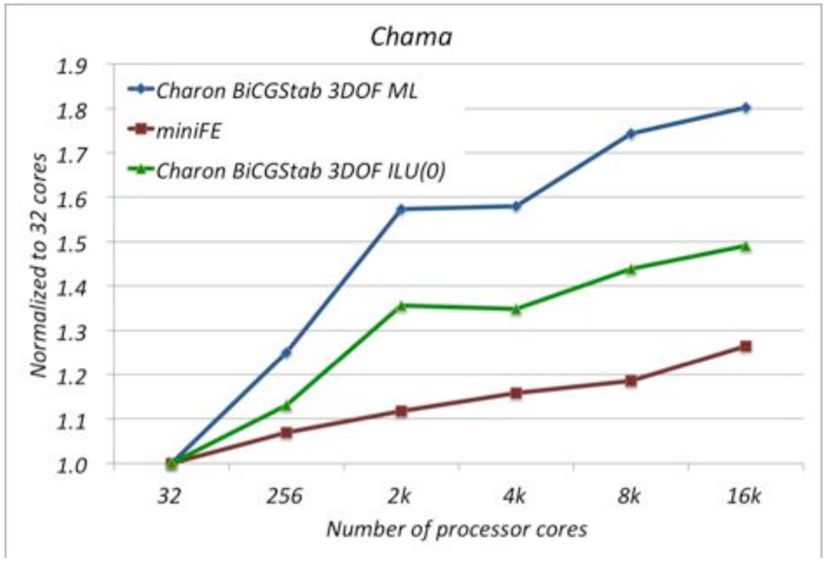


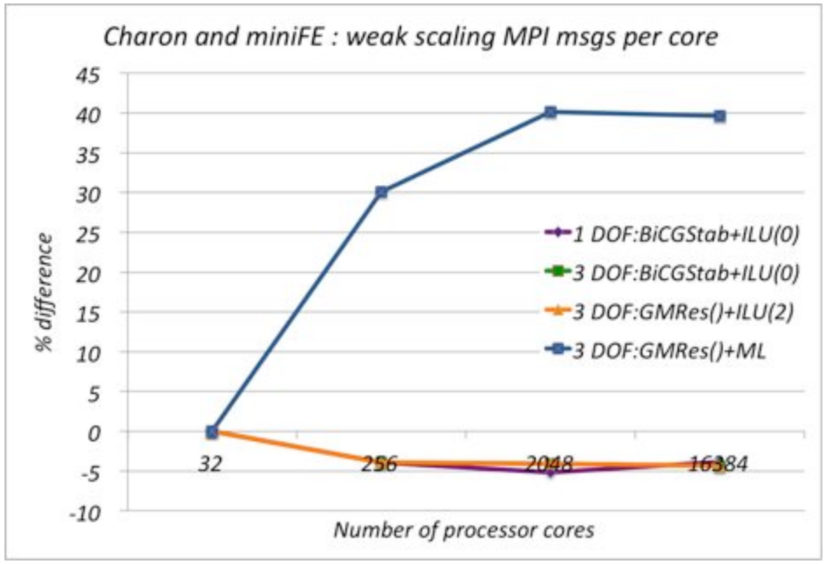
Charon and miniFE weak scaling

Diagnostic: Time for 1 iteration (min), normalized for each to 32 cores









Charon/miniFE Conclusions

- Data was collected on Chama, Cielo, Red Sky, workstations.
- Metrics investigated included on-node memory bandwidth, cache performance, and weak scaling.

We find that

- Matrix assembly is not impacted by memory bandwidth.
- Sparse iterative solver is impacted by memory bandwidth.
- Solver cache performance predictive; not for matrix assembly.
- miniFE does not capture behavior of multigrid preconditioner.
- Unclear if miniFE captures scaling behavior of other Charon solvers.

CTH and miniGhost
Halo Boundary exchange

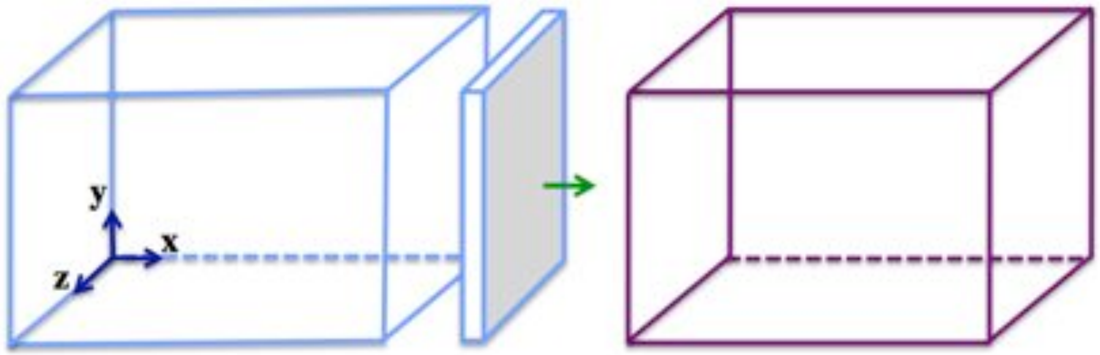
- Structured mesh, stencil computations, boundary exchange with message aggregation across many variables
  - Multi-MByte messages exchanged with (up to) 6 nearest neighbors.

- Goal : Effective inter-process communication strategies, now and in future.

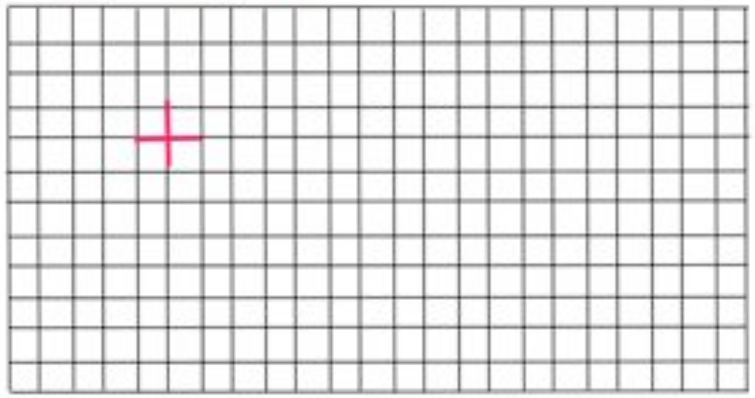
- Diagnostics:
  - Communication pattern: consistency requirement.
  - Scaling.

CTH and miniGhost Boundary exchange

```
DO I = 1, NUM_VARS
```



```
END DO
DO I = 1, NUM_VARS
```



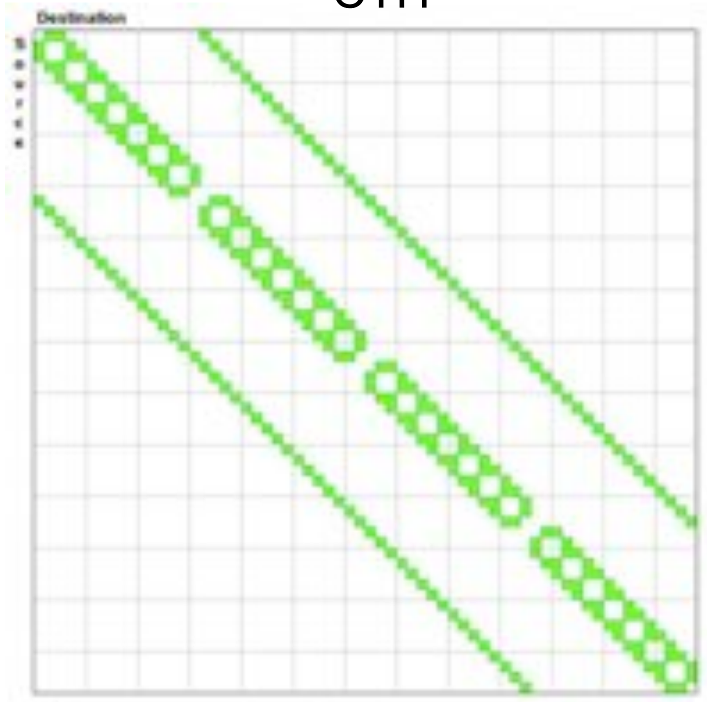
```
END DO
```

CTH and miniGhost :

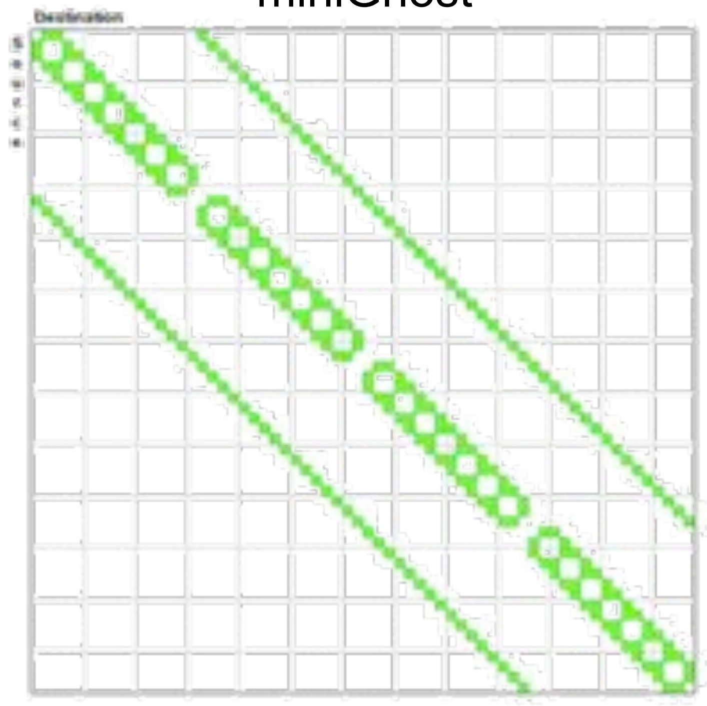
Diagnostics: Communication patterns and message sizes

miniGhost: Number of pt-2-pt neighbors and message size configured to match CTH problem sets.

CTH

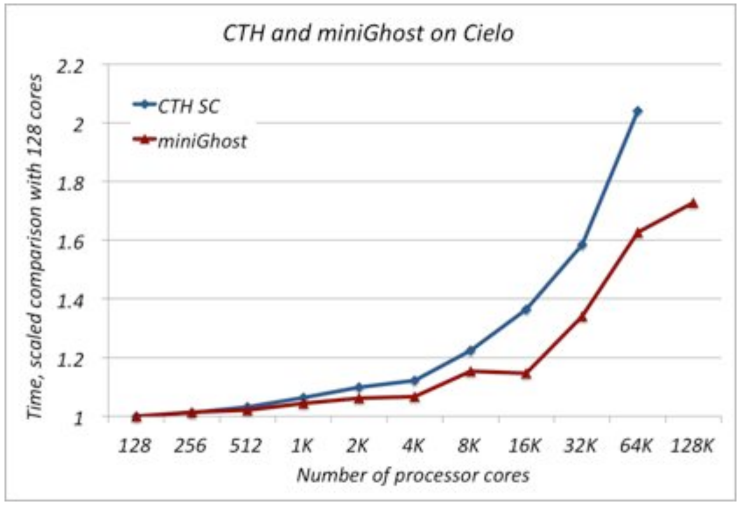


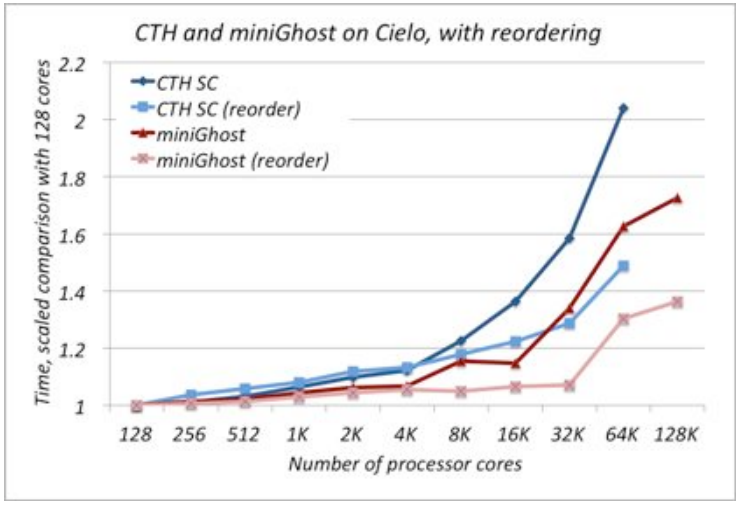
miniGhost



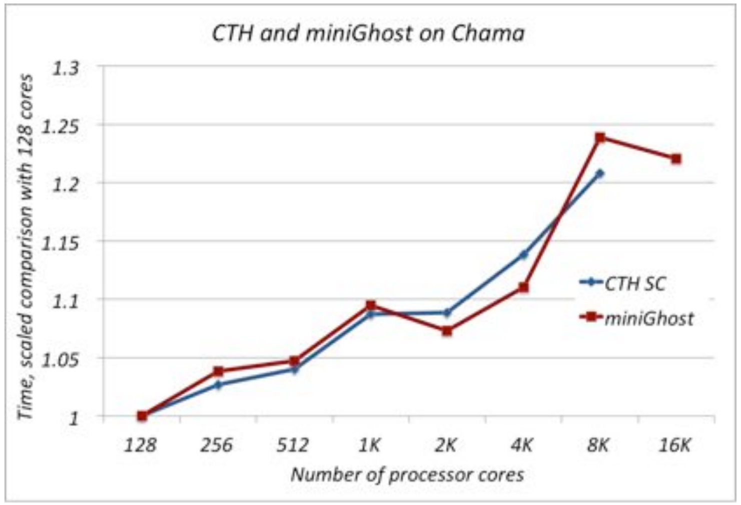
CTH and miniGhost :
Diagnostic: weak scaling, relative to 128 cores

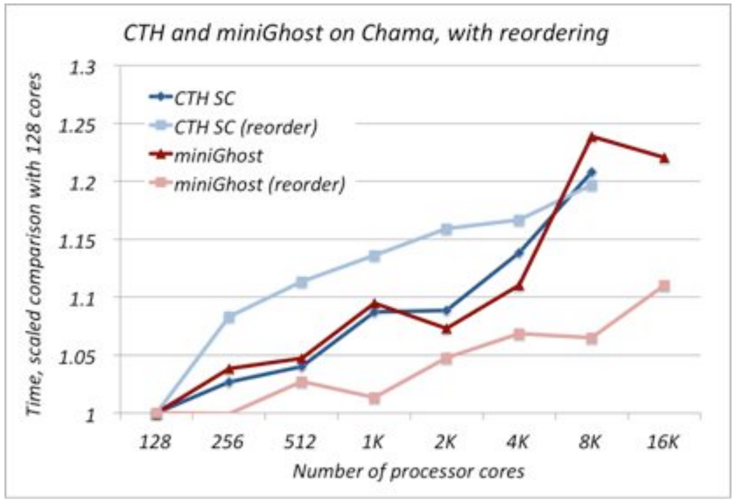
Cielo



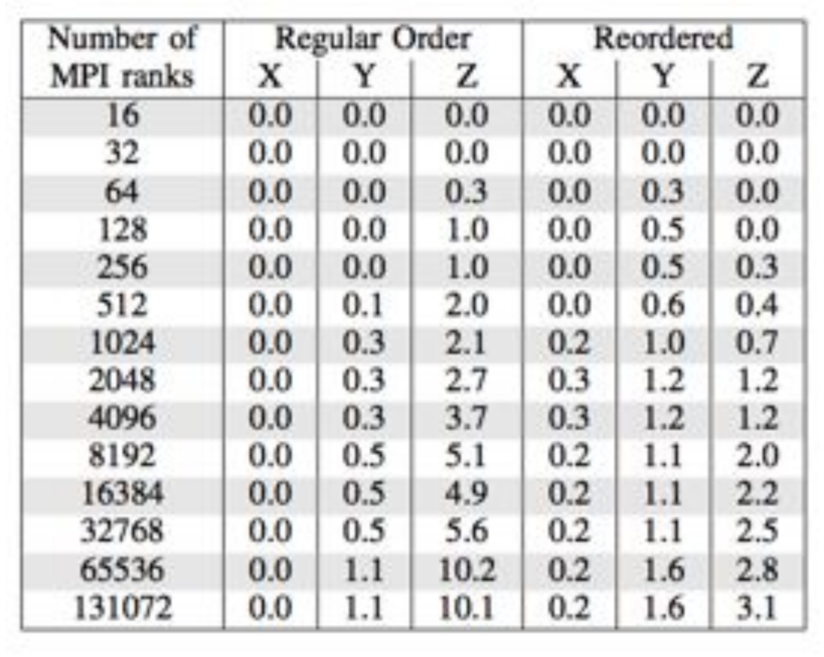


Chama



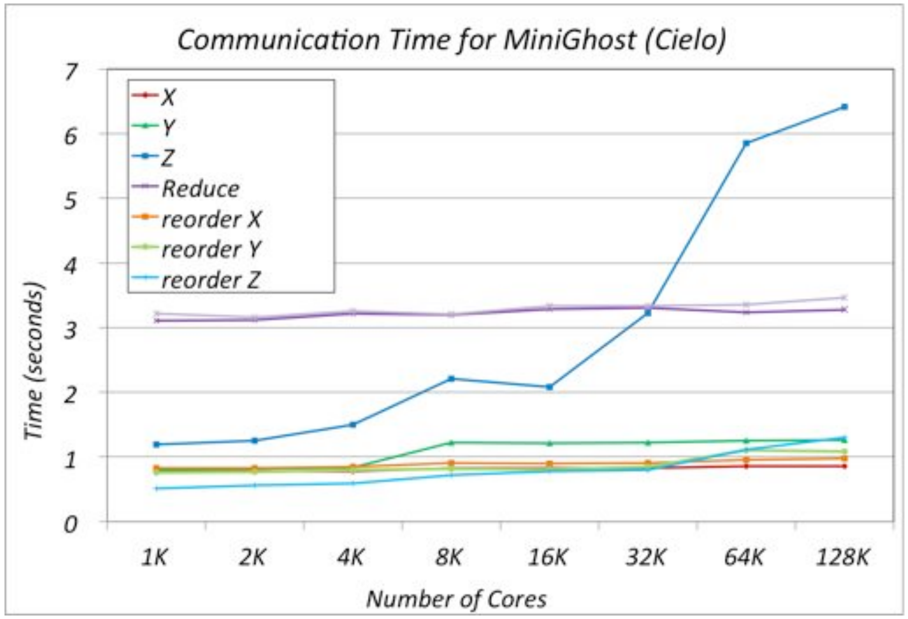


CTH and miniGhost: Diagnostics: Number of hops and communication costs per direction



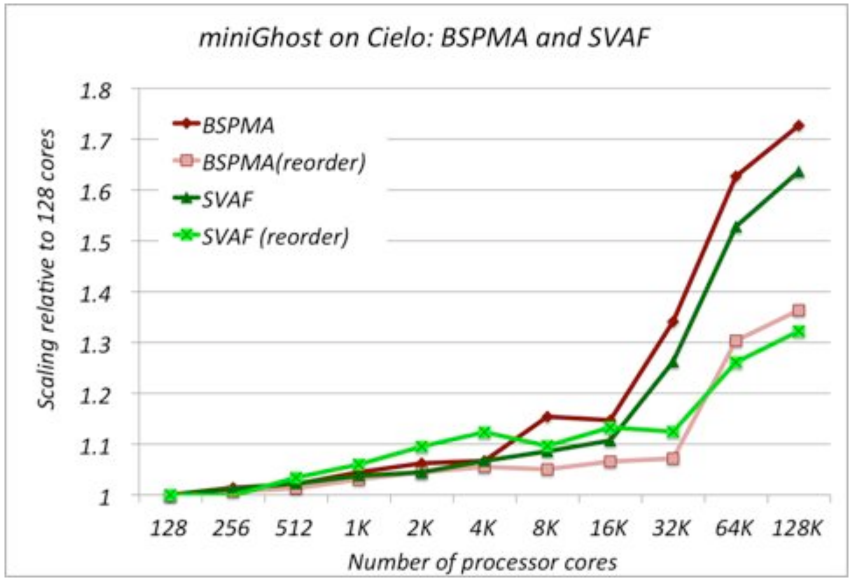
Number of MPI ranks	Regular Order X	Regular Order Y	Regular Order Z	Reordered X	Reordered Y	Reordered Z
16	0.0	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0	0.0
64	0.0	0.0	0.3	0.0	0.3	0.0
128	0.0	0.0	1.0	0.0	0.5	0.0
256	0.0	0.0	1.0	0.0	0.5	0.3
512	0.0	0.1	2.0	0.0	0.6	0.4
1024	0.0	0.3	2.1	0.2	1.0	0.7
2048	0.0	0.3	2.7	0.3	1.2	1.2
4096	0.0	0.3	3.7	0.3	1.2	1.2
8192	0.0	0.5	5.1	0.2	1.1	2.0
16384	0.0	0.5	4.9	0.2	1.1	2.2
32768	0.0	0.5	5.6	0.2	1.1	2.5
65536	0.0	1.1	10.2	0.2	1.6	2.8
131072	0.0	1.1	10.1	0.2	1.6	3.1

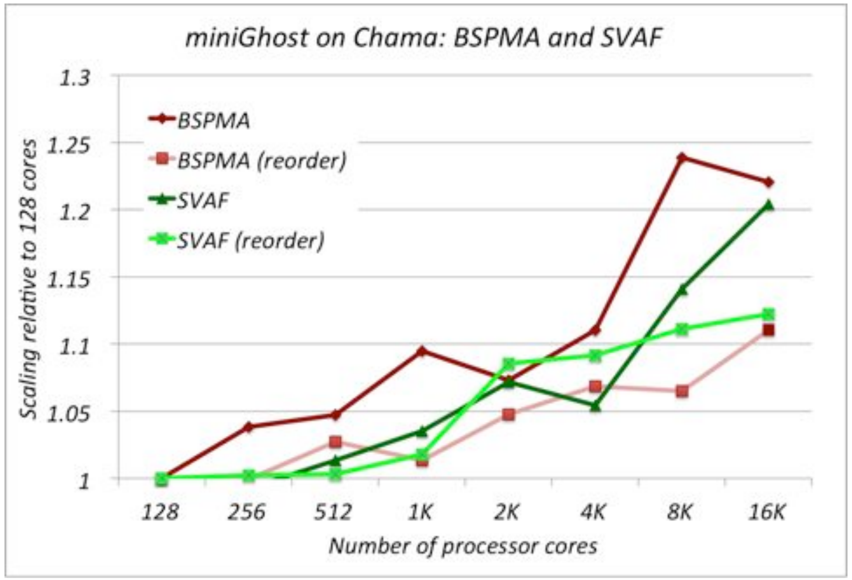
MINIGHOST AVERAGE HOP COUNTS ON CIELO



miniGhost: Alternative communication strategy

Diagnostic: weak scaling, relative to 128 cores





Profiling shows that computation time remains constant, but need more experiments to make stronger claim.

CTH/miniGhost Conclusions

- Data was collected on a Chama, Cielo, and Red Sky.

We find that

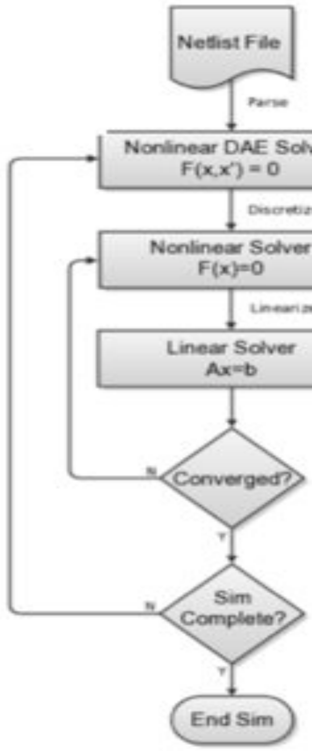
- miniGhost captures performance of inter-process boundary exchange,
- miniGhost informed effective process re-ordering strategy for CTH, and
- miniGhost suggests an effective alternative strategy for higher processor counts.

Xyce and miniXyce Electronic Circuit Simulation

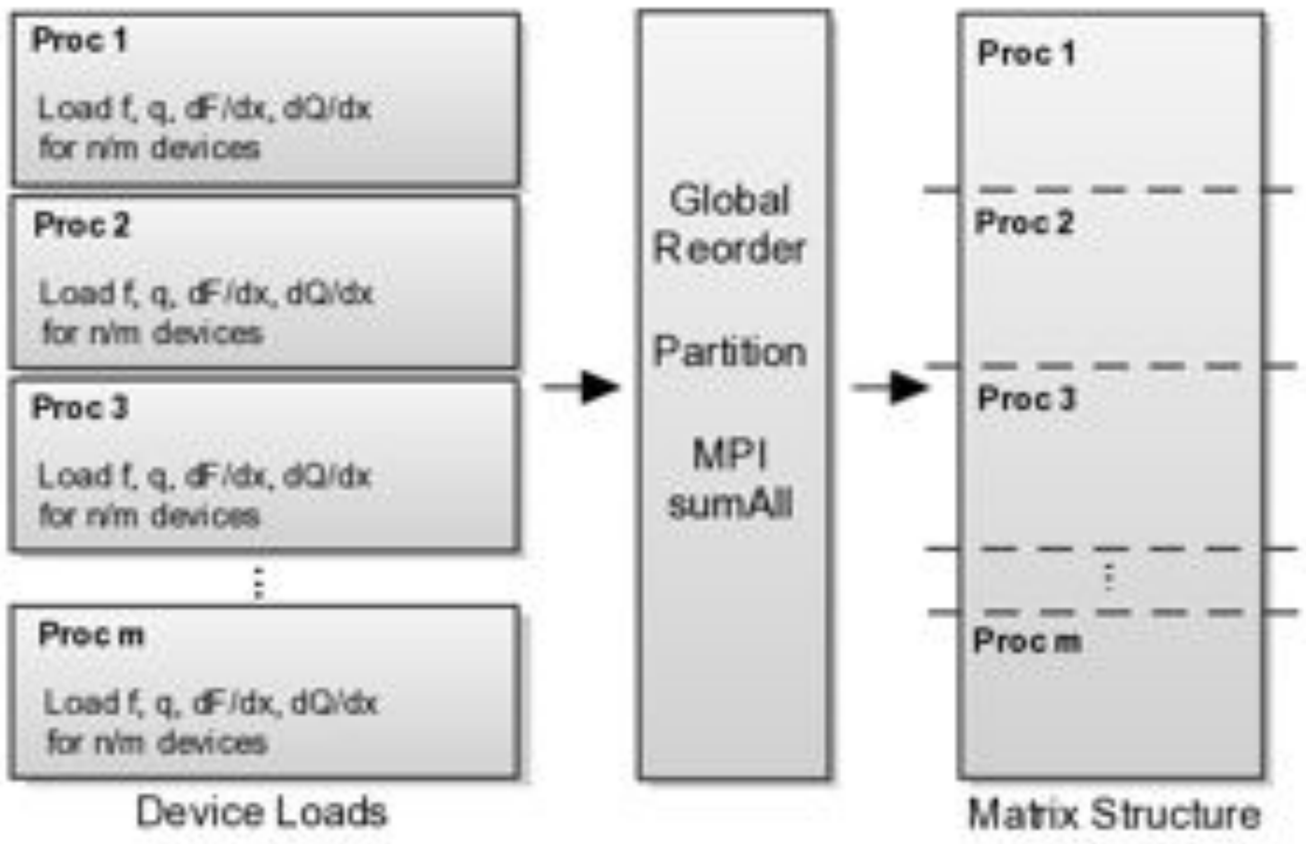
- Through this process the Xyce team realized miniXyce is not designed to capture the key performance issues important for exascale.
  - *miniXyce.v1 is a compact app.*
- Using our methodology to enhance miniXyce.

Electronic Circuit Simulation

- Transistor-level simulations for extremely large scale computers.
- Goal : Understand problem setup.
  - Circuit network partitioning
  - Device load balance
- Diagnostics:
  - Induced inter-process communication
  - Balanced computational workload
- Metric: proportional difference in time, scalability, etc.
- Linear solver: new miniapp for hybrid methods?



Xyce: Load balancing


Proc 1
Load f, q, dF/dx, dQ/dx for n/m devices
Proc 2
Load f, q, dF/dx, dQ/dx for n/m devices
Proc 3
Load f, q, dF/dx, dQ/dx for n/m devices
Proc m
Load f, q, dF/dx, dQ/dx for n/m devices
Device Loads
Global Reorder
Partition
MPI sumAll
Proc 1
Proc 2
Proc 3
Proc m
Matrix Structure

Future plans

- All:
  - Investigate heterogeneous architectures.
  - Prepare for and inform Trinity procurement.
- miniFE
  - Investigate ML preconditioning.
- miniGhost
  - Apply reorder to CTH with AMR
  - Alternative boundary exchange strategies, including PGAS, etc.
  - Increase relevance of computation.
  - Inform development of general process mapping strategies.

Text of Milestone

Description: The Mantevo project includes a set of application proxies, referred to as "mini-apps," and designed by code developers to represent key runtime performance characteristics of their applications. SNL will analyze two of these mini-apps to determine how well they represent the full application programs. Specifically, SNL will profile the runtime performance of the mini-app and application, characterizing the relationship between the two on at least two HPC platforms (including Cielo).

Contribution to Stockpile Stewardship: The long term usefulness of ASC codes to the Stockpile Stewardship program will necessitate that the codes evolve as supercomputer architectures change. Dramatic changes to full production codes are expensive and high risk. Applications proxies can play an important role in determining the evolutionary path of production codes without incurring the high overhead of working with the full code.

Summary

- Defined a methodology, providing a framework for reasoning about key performance issues in application codes.
- Applied this to a set of applications and miniapps.
- Middle of the beginning.
  - Continue building a body of evidence.
  - Expect methodology to refine, change, etc.
- L2 FY13 (tentative): "*Study of Key performance Issues of ASC Applications Executing on Emerging Technologies*": demonstrate predictive capabilities