Characterizing Compiler Performance for the AMD Opteron Processor on a Parallel Platform

Douglas Doerfler
Courtenay Vaughan
Sandia National Labs

May 17th, 2005
CUG 2005

Introduction

- Cray & Sandia National Labs (SNL) have chosen the Portland Group (PGI) C, C++ and Fortran compiler suite as the nominal compilers for SNL's Red Storm

- This study takes a look at the Pathscale compiler suite by comparing it against the PGI suite on applications of interest to SNL and the ASC program

- There are other excellent Opteron compilers available, but this study is an outcome of a collaboration with Pathscale to help the maturation of their product
  - However, neither company was consulted for tuning advice
  - The companies respective literature and white papers were consulted

- In addition to single processor performance, we looked at scaled (weak-scaling) problem sets and make inferences about the applications ability to hide parallel overheads and hence take advantage of compiler optimizations at scale

Fractional Speedup

- Total fraction of time before optimization is defined as

$$F(N) = f_c(N) + f_o(N) = 1$$

  - where f_c is computational fraction and f_o is the fraction of overhead
- Total fraction of time after optimization is defined as

$$F_s(N) = f_c(N) * f_s(N) + f_o(N) < 1$$

  - Where f_s is the fractional speedup
- F_s is measured by the application and is defined as

$$F_s(N) = (\text{time for faster runtime}) / (\text{time for slower runtime})$$

- And percent speedup is defined as

$$\text{Speedup (\%)} = (1 / F_s(N) - 1) * 100$$

Environment

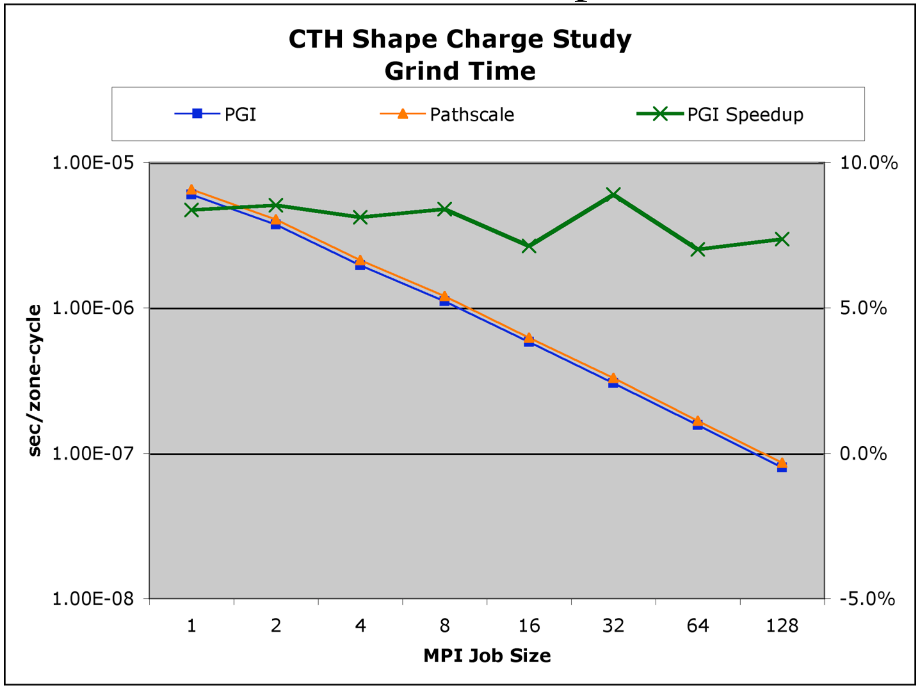
- PGI Compiler Suite Version 6.0
  - -fastsse
    - -fast -Mvect=sse -Mscalarsse -Mcache_align –Mflushz
    - -fast -> -O2 -Munroll=c:1 -Mnoframe –Mlre
- Pathscale Compiler Suite Version 2.1
  - -O3 -OPT:Ofast
    - -OPT:Ofast -> -OPT:ro=2:Olimit=0:div_split=ON:alias=typed -msse2
- Red Squall Cluster
  - Dual-Processor 2.0 Ghz Opterons
  - 333 Mhz DDR DRAM
  - Quadrics QsNetII High-Speed Interconnect
  - MPICH 1.2.4 w/Quadrics extensions
  - SuSE Linux Professional 9.0 on all nodes

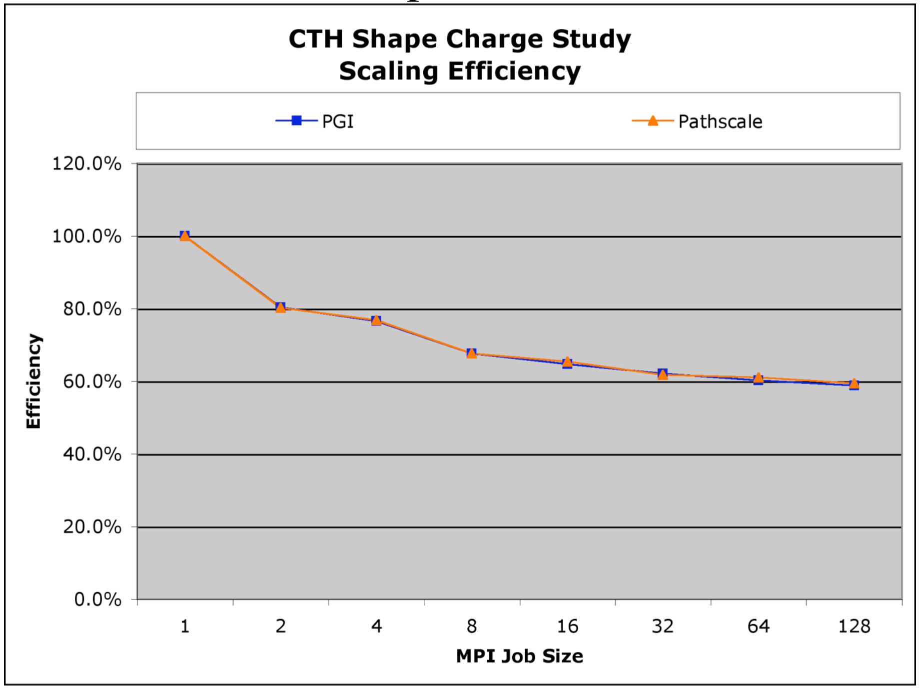
Applications

- CTH
  - Multi-material, large deformation, strong shock wave, solid mechanics code
  - Fortran 77
  - Chosen for its extensive use at SNL and its characteristic 3-D Mesh message passing traits
- LAMMPS
  - (Large-scale Atomic/Molecular Massively Parallel Simulator) is a classical molecular dynamics code designed for simulating molecular and atomic systems on parallel computers using spatial-decomposition techniques
  - Fortran 90
  - FFTW is implemented in C
  - Chosen for its extensive use at SNL and well characterized behavior
- PARTISN
  - LANL Developed Code
  - Neutron transport solutions on orthogonal meshes with adaptive mesh refinement (AMR) in one, two, and three dimensions
  - Chosen for its irregular memory transfer characteristics

CTH

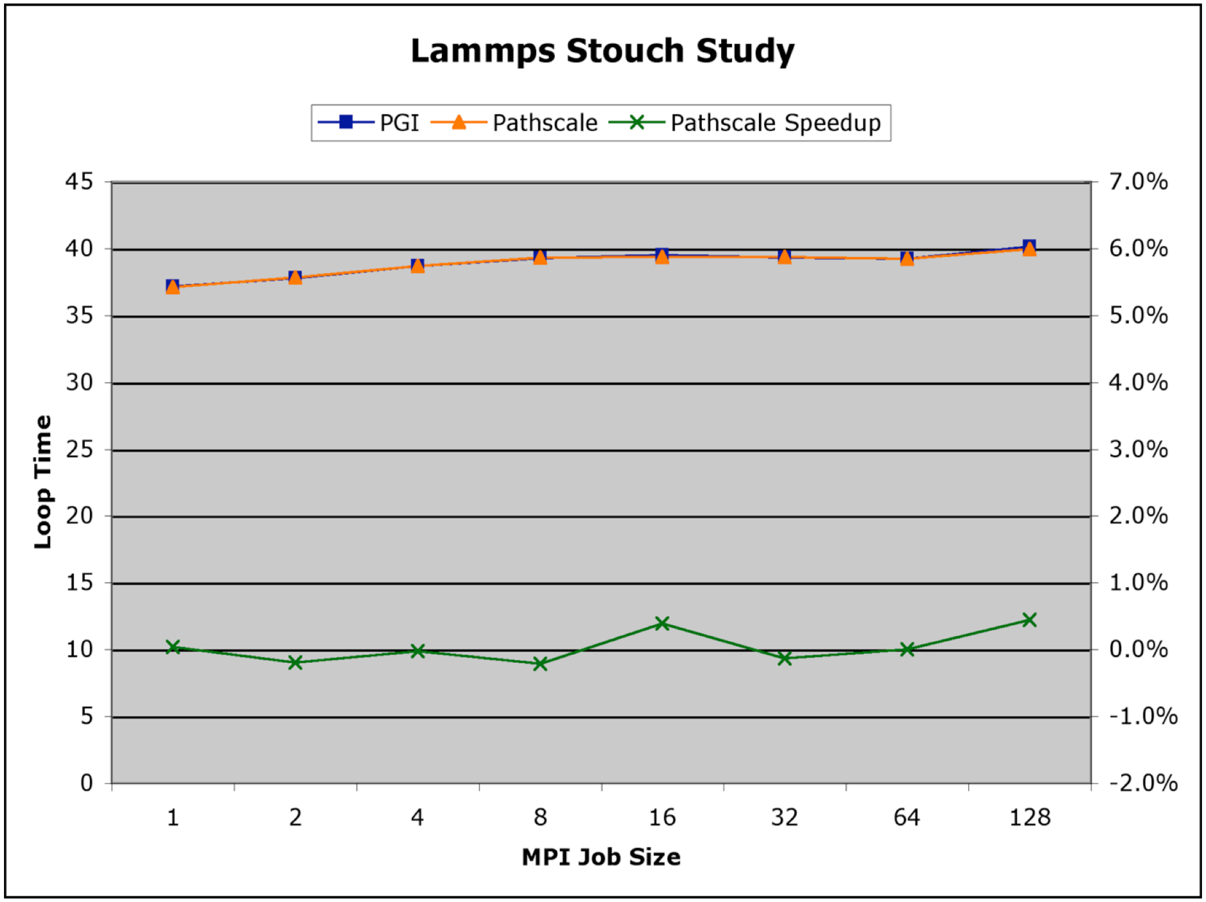
- Shape Charge problem set derived from a real calculation.
- Reports "grind time", which halves as the problem doubles in size.
- Unable to use -O3 on Pathscale compiler, so -O2 -OPT:Ofast was used.
- Relatively constant advantage for the PGI compiler as the problem scales. Indicates that f_c and f_o remain constant with scale.
- This implies that the parallel inefficiency is due to algorithmic issues and that the total time for computation and overhead increases as the problem scales.





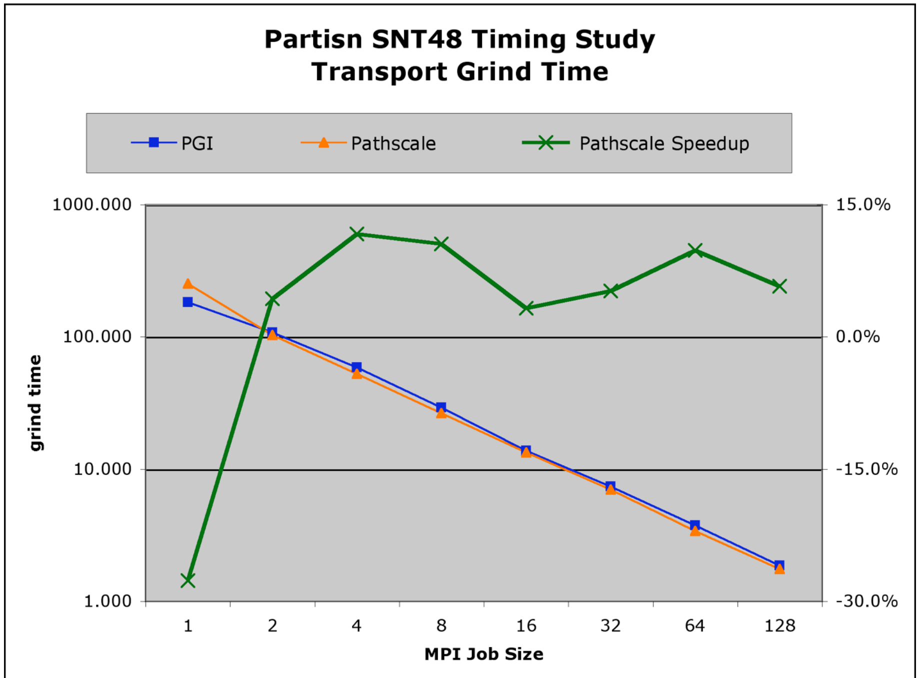
LAMMPS

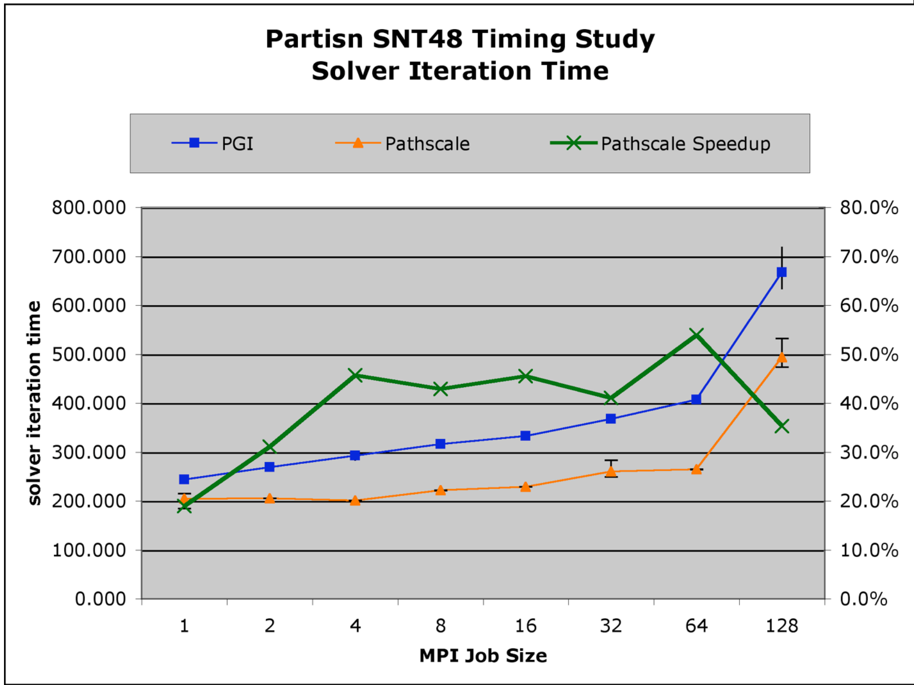
- Indifferent as to which compiler to use
- Unable to characterize applications ability to take advantage of compiler optimizations

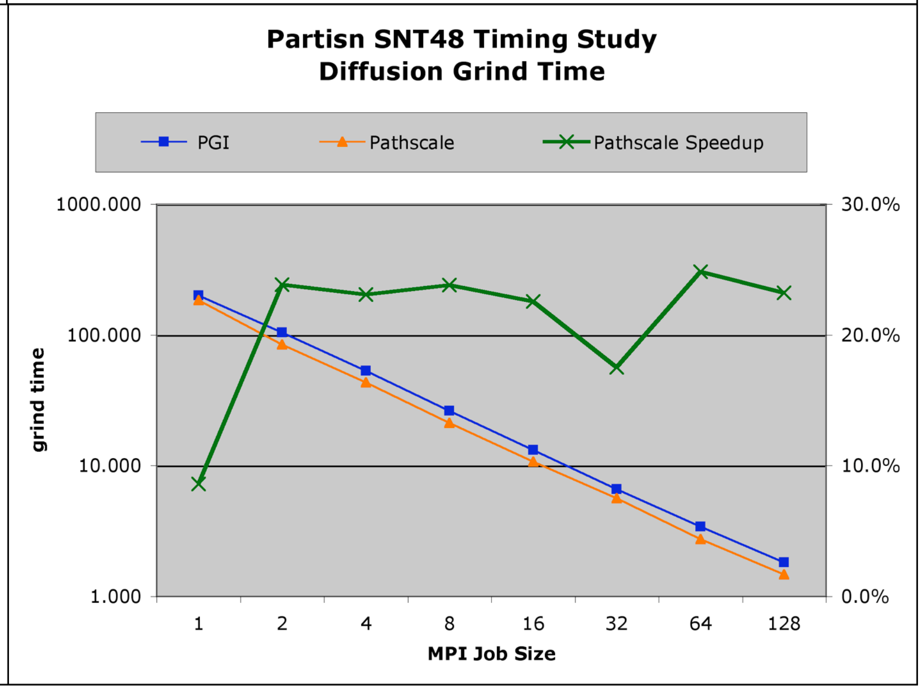


PARTISN

- Three metrics measured
  - – Transport Grind Time
  - – Diffusion Grind Time
  - – Solver Iteration Time
- Single node anomaly for Pathscale result
- Pathscale advantage from 2 to 128 nodes
  - – Overhead ~ constant as scale increases
- Solver Iteration result at 128 nodes may be due to the Red Squall network architecture







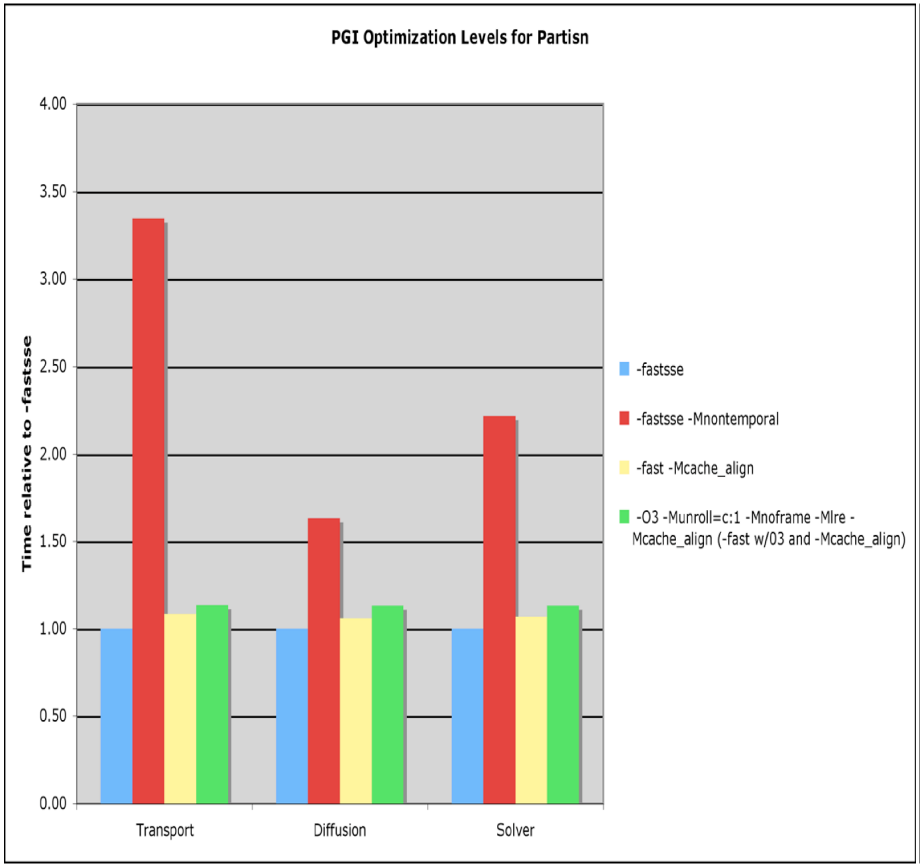
Different Optimization Settings for PARTISN on a Single Node

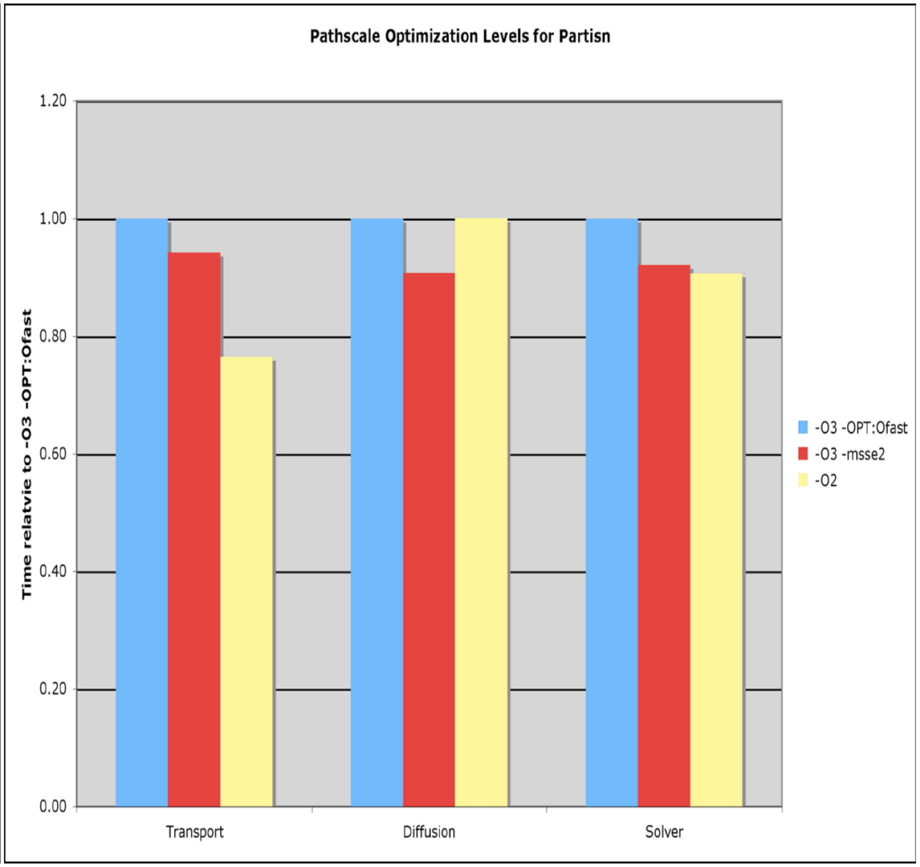
PGI

- -fastsse
- -fastsse -Mnontemporal
- -fast -Mcache_align
- -O3 -Munroll=c:1 -Mnoframe -Mlre -Mcache_align (-fast w/03 and -Mcache_align)

Pathscale

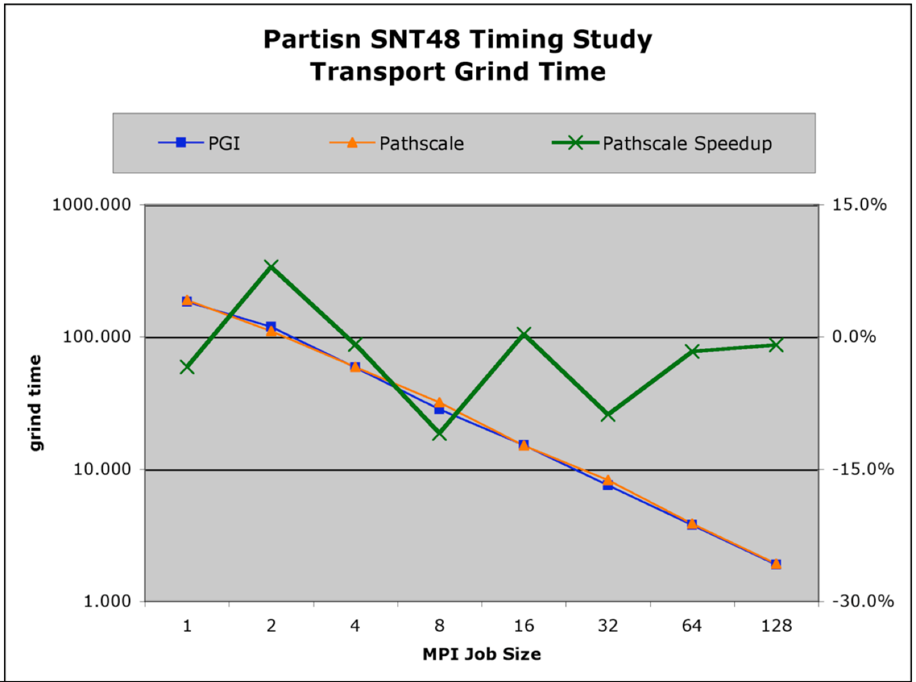
- -O3 -OPT:Ofast
- -O3 -msse2
- -O2

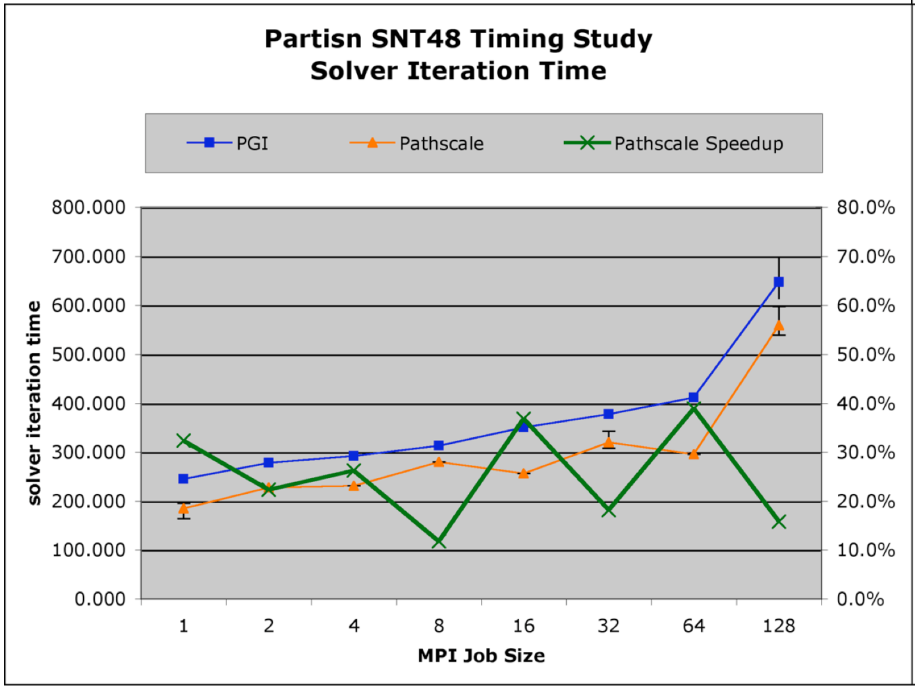


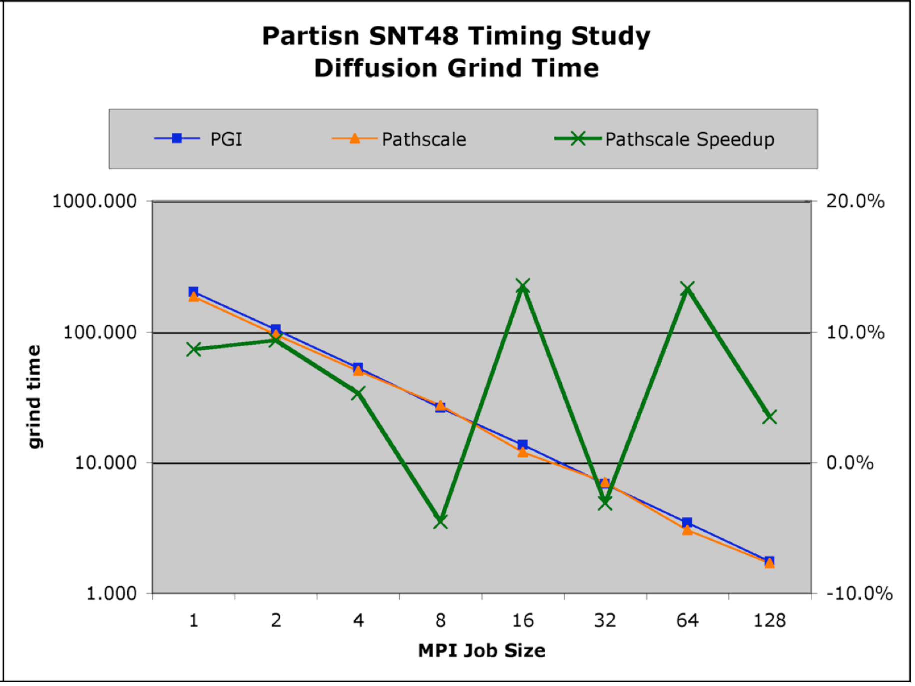


PARTISN Continued

- Used -O2 with Pathscale
- Left PGI at -fastsse
- Improved Pathscale single node performance
- However, overall Pathscale performance advantage reduced
- Performance dependent on scale
  - –8, 32 and 128 node anomalies (?)







Summary and Conclusions

- The PGI Compiler provided the best results for CTH.
- The Pathscale Compiler provided the best results for PARTISN.
- LAMMPS was indifferent.
- Each ended the season with a record of 1-1-1.
- Both compilers can be tweaked with numerous switches and it is most likely possible that excellent run time results can be obtained with either compiler.
- However, it seems beneficial to the application developer to have multiple compilers to choose from, as one may provide a more optimal result for a given application without significant “fishing” for the right options.
- Compilers can be a significant monetary investment, but after taking into account increased efficiency of the platforms resources over an extended period of time, it may be worth it.

Questions?