SAND2006-3933C

Performance Modeling of CTH Run Times on MPP and Cluster Platforms

DoD High Performance Computing Modernization Program

2006 Users Group Conference

Denver, CO

June 26-29, 2006

Sue P. Goudy, SMTS

Sandia National Laboratories

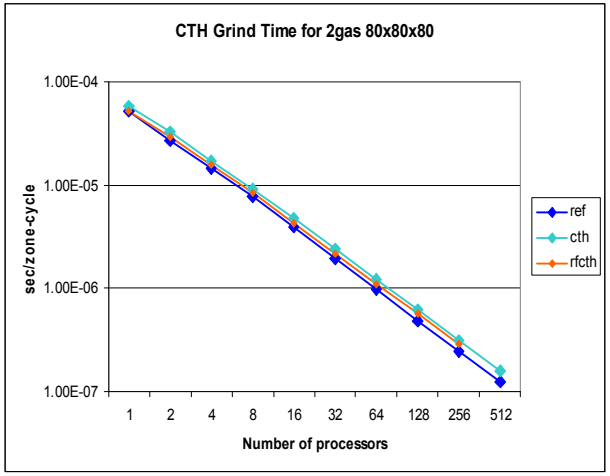
Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

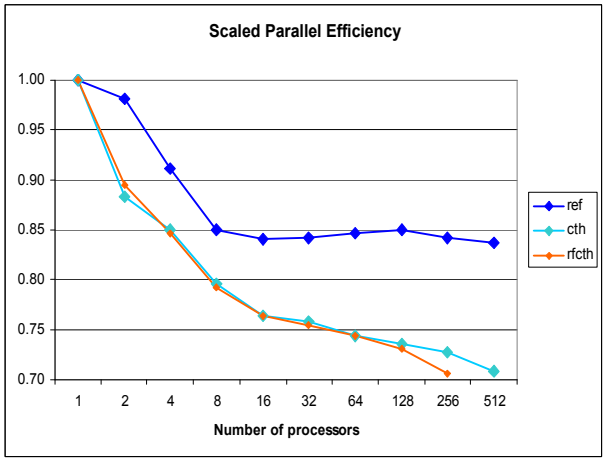
Acknowledgments

- To the Performance Analysis Team at Sandia for timing data (and a viewgraph or two)
  - Especially Courtenay Vaughan, Mahesh Rajan, and Bob Benner
- To Steve Schraml for input decks, both fixed mesh and automatic mesh refinement
- To my previous and current managers at Sandia for supporting this work

CTH Scaling Studies

- Data taken from ASCI Red scaling studies, 2004
- Comparison of production CTH (Int03) and rfcth with reference CTH (pre-AMR)
- Upper chart: familiar presentation of grind times on log-log scale
- Lower chart: weakly scaled parallel efficiency, derived from grind times
- Reasonable scaling for Int03, better scaling for reference version – WHY?





Sandia National Laboratories

Basis of Model

- Computational complexity of $O(N^3)$ where N is the length of one edge of a processor's subdomain
- Communication complexity for the data exchanges is $O(N^2)$
- Communication complexity of collective operations is $O(log(P))$ where P is the number of processors

A Model of CTH (flat mesh)

$$T = E(\kappa,\varphi)N^3 + C(\lambda + \tau k N^2) + S(\gamma\, log(P))$$

- T is the time per time step
- N is size of an edge of a processor's subdomain
- P is the number of processors
- C and S are number of exchanges and collectives
- *k* is the number of variables in an exchange
- λ and τ are latency and transfer cost
- γ is the cost of one stage of collective
- $E(\kappa,\varphi)$ is the calculation time per cell

Notes for Semi-empirical Model

- Number of halo exchanges per time step varies
  - Approximately 20 for 2 processors
  - More than 100 for a processor that communicates with all possible neighbors (six in flat-mesh mode)
- Communication profiles record 89 MPI_Allreduce transfers per time step

- Parameters λ, τ, γ are machine dependent
- Used Pallas benchmarks on RedStorm
  - PingPing values λ=8.3µs , τ=0.00102 µs/byte
  - Allreduce value γ=2.6 µs/byte

- Parameter pair κ,ϕ is hard to quantify because operation count and effective floating point performance depend on code phase
- Practice is to use a measured value, typically derived from single processor execution time, to represent E(κ,ϕ)

Model vs. Measured execution time

Processors	Time per Time Step	Model
1	11.83	11.83
2	14.23	11.94
4	14.86	11.94
8	17.17	12.05
16	17.49	12.05
32	18.70	12.05
64	18.86	12.05
128	19.73	12.27
256	19.86	12.27
512	21.95	12.27
1024	22.01	12.27
2048	22.16	12.27
4096	22.10	12.27

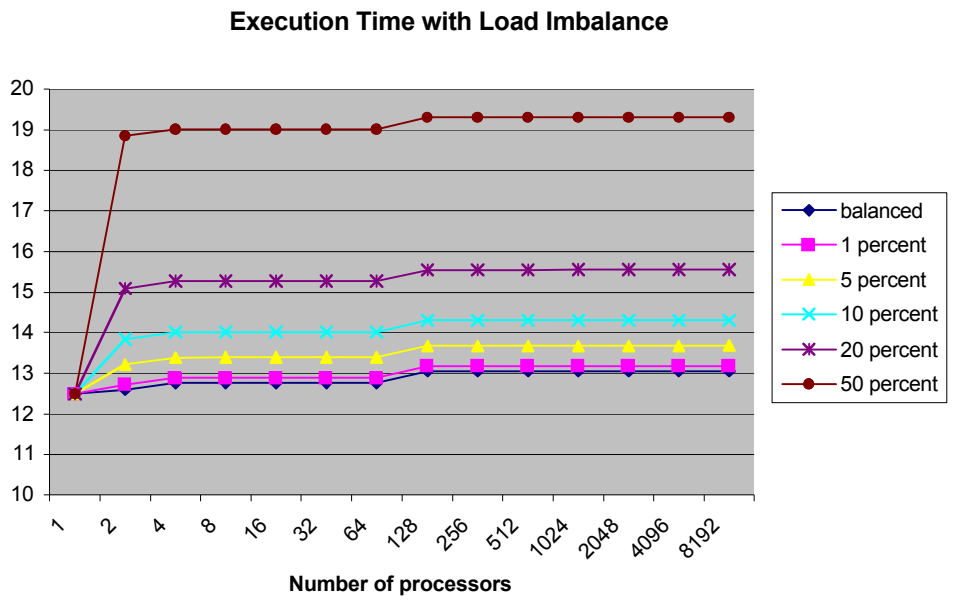
Typical load imbalance portrait from Intel Trace Collector

Sandia National Laboratories

CTH Model with Load Imbalance

$$T = L_{imbal}\, E(\kappa,\varphi)N^3 + C(\lambda + \tau k N^2) + S(\gamma\, log(P))$$

- T is the execution time per time step
- N is size of an edge of a processor's subdomain
- C and S are number of exchanges and collectives
- P is the number of processors
- *k* is the number of variables in an exchange
- λ and τ are latency and transfer cost
- γ is the cost of one stage of collective
- $E(\kappa,\varphi)$ is the calculation time per cell
- L_{imbal} is a new term representing effects of load imbalance



- Term L_{imbal} depends on several factors
  - Problem size, dimension, and data decomposition across processors
  - Number of processors and their mapping to the data decomposition
  - Assignment of special tasks (e.g. writing log files) to one processor
- Quantification approach
  - Define a weak scaling study on the simulation of interest for a small number of processors, say 1-16.
  - Use a trace collector to measure communication imbalance over the entire simulation; derive the load imbalance for user code.
  - Determine a statistical metric for load imbalance, say the mean of the ratios of the maximum local time for user code to the average local time for user code.
  - Use this metric as the value of L_{imbal} in the execution time model.

Sandia National Laboratories

Work in progress

- Quantification of load imbalance
  - Uses mympic.c (U. Pittsburg) trace collection library and a set of text post-processing tools
  - Currently examining simulations that have known load imbalance issues (e.g. shape charge)
- How does load imbalance vary with number of processors?
  - 2:1.01 4:1.05 8:1.21 16:1.23 32:?
- How does load imbalance vary with computing platform?
- And does code phase make a difference?

For more information

- Vaughan, Courtenay T., Sue P. Goudy, "Analysis of an Application on Red Storm," Cray Users Group, May 2006.

- spgoudy@sandia.gov

Sandia National Laboratories

Abstract

Predicting CTH run times is extremely difficult because the actual run time activities within the simulation depend strongly on input specifications. Therefore, a performance model for CTH must take into account the particular problem being simulated in order to determine which computational phases occur and the space/time complexity of these phases. Part of the challenge in creating of a good execution time model for CTH lies in the number of ways that computational phases can be organized and combined. The number of materials and their arrangement in the mesh form the basis of complexity analysis for a performance model. The variety of options available for equation-of-state and material strength calculations that are done during the Lagrangian step complicate the modeling process. There are also options for interface tracking and methods of handling cells with multiple materials that introduce variations in numerical intricacy. Moreover, the sequence of computational phases for a time step can change during simulation, depending on material discard and/or fracture models.

The focus of this paper is an experimental methodology for obtaining bounds on the variability of per-processor and per-cycle execution time for CTH. Results of applying the technique on a typical benchmark simulation for CTH, in the context of different domain decompositions, are presented. The dynamic runtime information from these experiments is used to enhance a statically determined performance model for CTH. Runtime data were collected from the Red Storm supercomputer and from large Linux clusters at Sandia National Laboratories.