SAND2015-1742C

Exploring Communication Options with Adaptive Mesh Refinement

Courtenay T. Vaughan, and Richard F. Barrett
Sandia National Laboratories

SIAM Computational Science & Engineering
March18, 2015

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.

CTH

- Three-dimensional shock hydrodynamics code
- In flat mesh mode, each processor has one large block and there are several points per timestep where each processor sends a few large messages aggregated from several variables per cell to up to six neighbors
- In AMR mode, each processor has a number of smaller blocks and typically sends more smaller messages
  - Communication pattern change during run

miniAMR

- Stencil calculation code with AMR (Adaptive Mesh Refinement) – written in C
- Part of the Mantevo Suite (mantevo.org)
- Simple finite volume calculation
  - no real physics, but kernel could be easily modified
- Many smaller blocks per processor
- Each block does a halo exchange with neighboring blocks, which may be on processor or off
- Needs load balancing
  - One area may refine while rest does not
- Has complicated bookkeeping
  - each block has 6 to 24 neighbors and parent

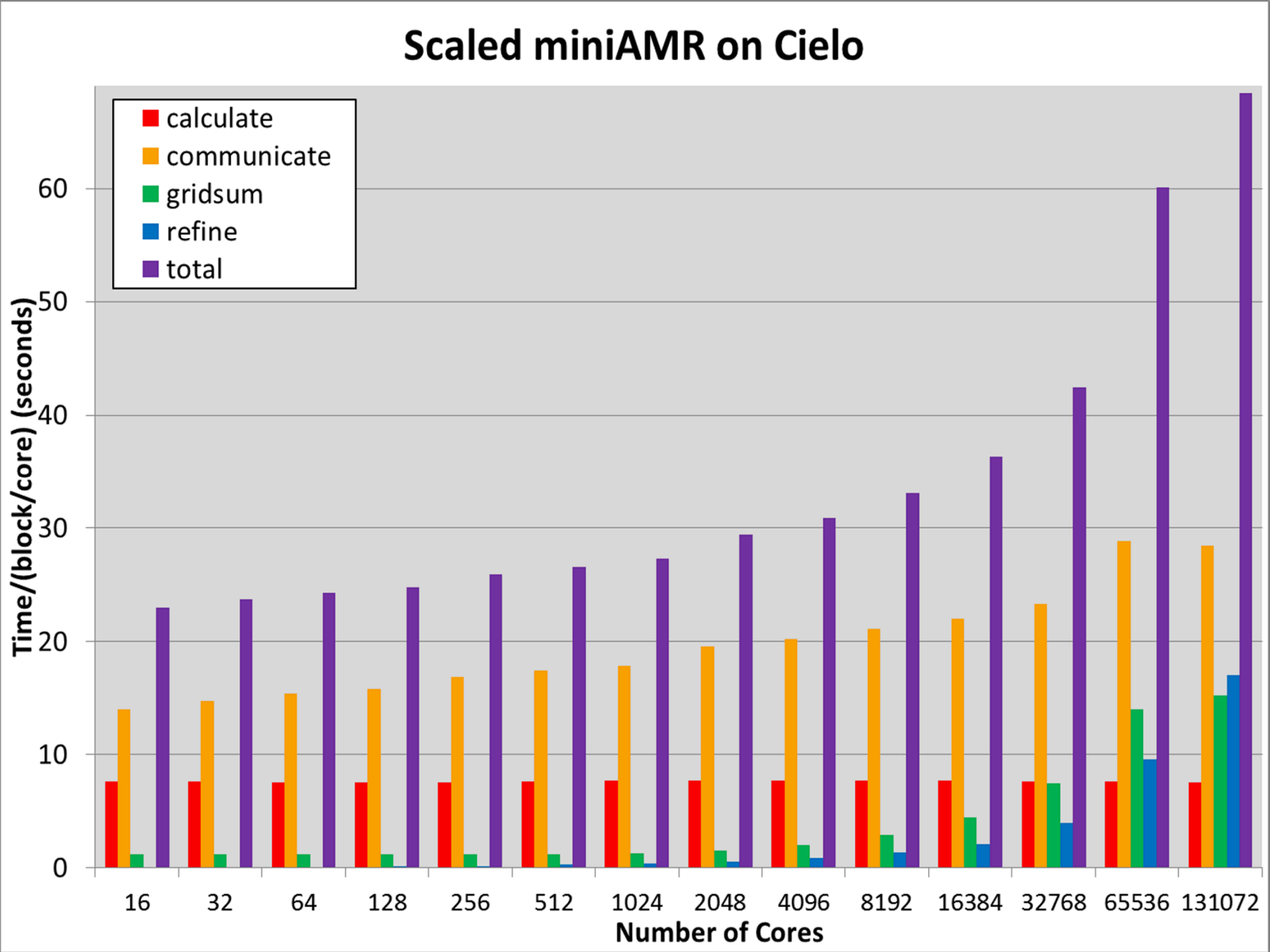
Comparison with CTH

- Run on 128 cores
- CTH problem is a sphere that hits a block at an oblique angle and produces a shock wave
  - modeled in miniAMR as a deforming spheroid with an expanding hemisphere to represent the shock
- CTH averages 140.9 blocks/core over the run
  - average core has 16.3 messages per communication stage that average 261 KB
- miniAMR averages 141.9 blocks/core over the run
  - average core has 18.4 messages per communication stage that average 224 KB

Comparison to CTH continued

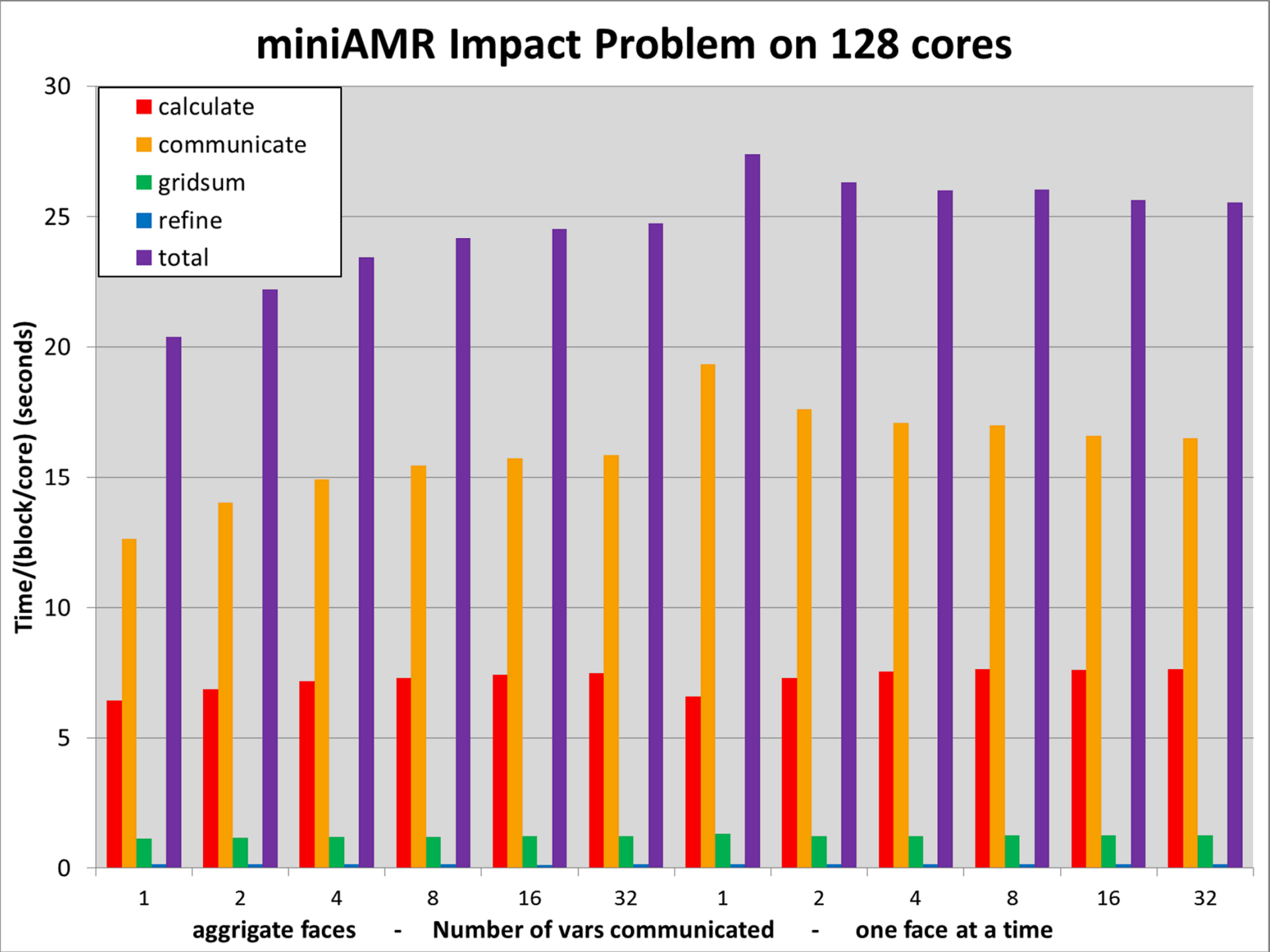
128 cores on Cielo	CTH	miniAMR
calculation	27.3%	35.4%
communication	61.5%	64.0%
refinement	11.2%	0.6%

In this comparison, communication includes just the communication of ghost cells between blocks. The calculation time includes calculation and communication necessary to calculate the timestep and other things using global reductions.


Scaled miniAMR on Cielo
calculate
communicate
gridsum
refine
total
Time/(block/core) (seconds)
0
10
20
30
40
50
60
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
Number of Cores

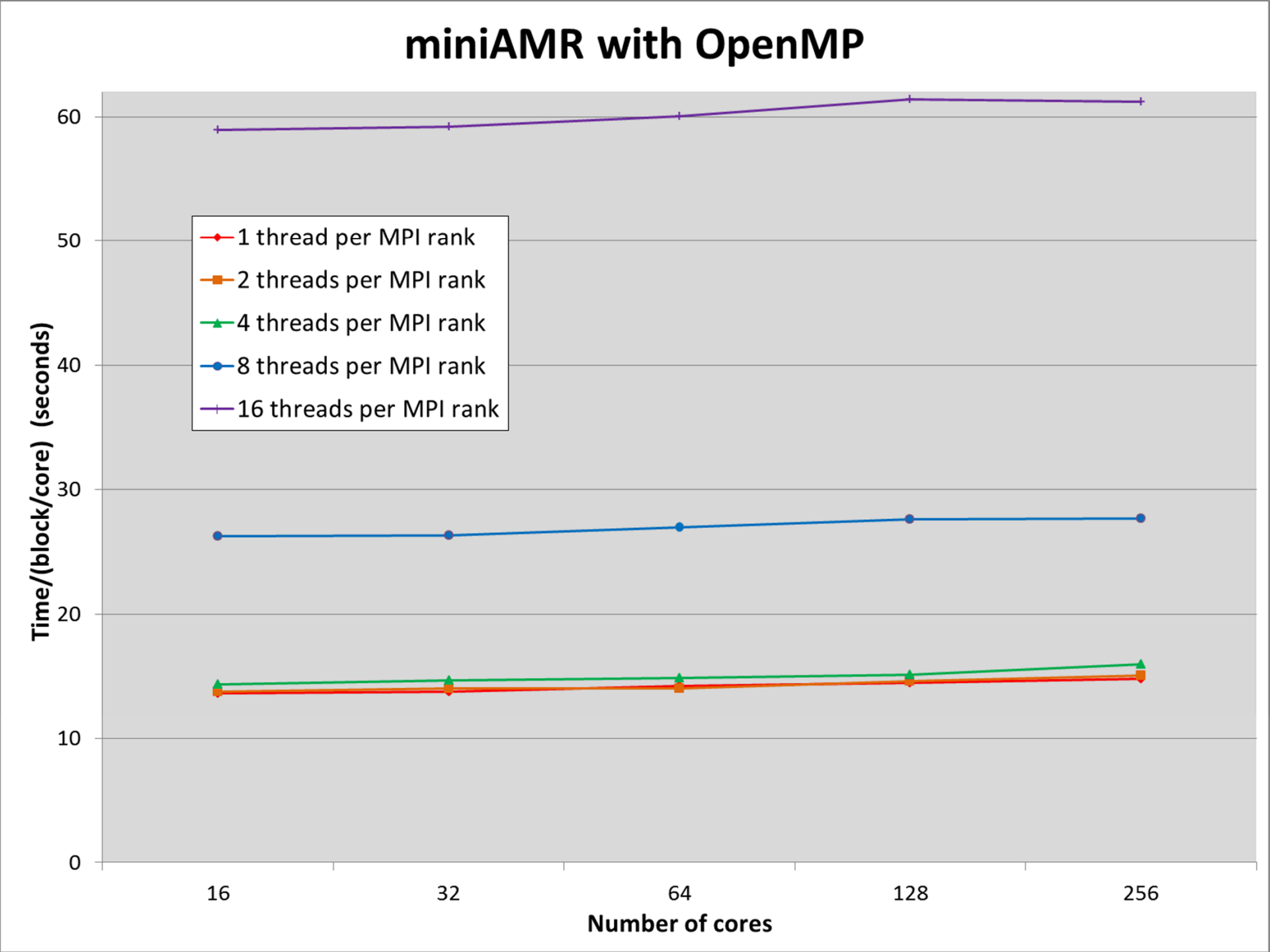
Comments on Scaled Speedup Curve

- Communication dominates the time and is increasing gradually
  - Includes time to communicate boundary information on blocks on the same core (30.6% of communication time on 128 cores)
- Calculation time is a consistent amount of time per block
  - If completely refined, then the 128 core problem would have 524288 blocks instead of 18168 and the calculation time would be 218 seconds instead of 7.6 seconds
- The refinement and gridsum times both are increasing gradually
- These reflect tradeoffs that AMR makes to allow problems to be run in less time on fewer nodes


miniAMR Impact Problem on 128 cores
calculate
communicate
gridsum
refine
total
Time/(block/core) (seconds)
0
5
10
15
20
25
30
1
2
4
8
16
32
1
2
4
8
16
32
aggrigate faces - Number of vars communicated - one face at a time

Comments on Chart

- Communication is usually done by sending all 32 variables at the same time
- The best performance is sending one variable at a time, which results in 32 times the number of messages than normal
- Sending one face at a time takes a bit longer with about 10 times as many messages as the aggregate faces method
  - Will be used as we try a task based decomposition


miniAMR with OpenMP
Time/(block/core) (seconds)
60
50
40
30
20
10
0
1 thread per MPI rank
2 threads per MPI rank
4 threads per MPI rank
8 threads per MPI rank
16 threads per MPI rank
16
32
64
128
256
Number of cores

Summary

- miniAMR was designed to study communication present in AMR codes such as CTH
- Experiments show that better performance can be achieved, to a point, with more smaller messages
- The OpenMP version shows reasonable scaling
  - Not all of the code is converted to OpenMP