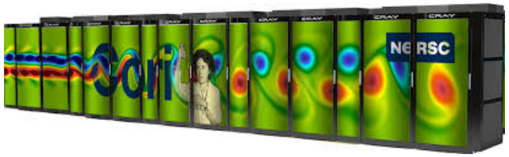


Chombo-Crunch case study

Andrey Ovsyannikov*
NESAP Postdoc, NERSC

with David Trebotich and Brian Van Straalen (CRD, LBL)

*Email: aovsyannikov@lbl.gov

November 3, 2016

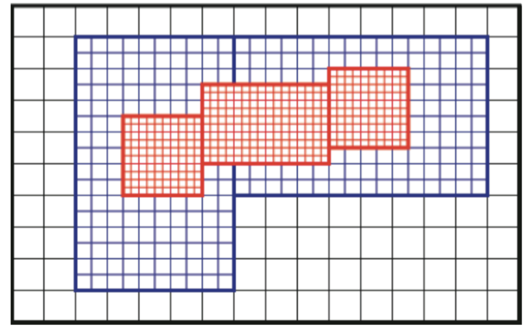
Chombo-Crunch

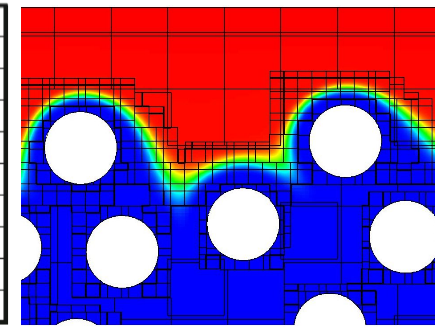
CFD + multi-component geochemical reactive transport in very complex pore scale geometry:

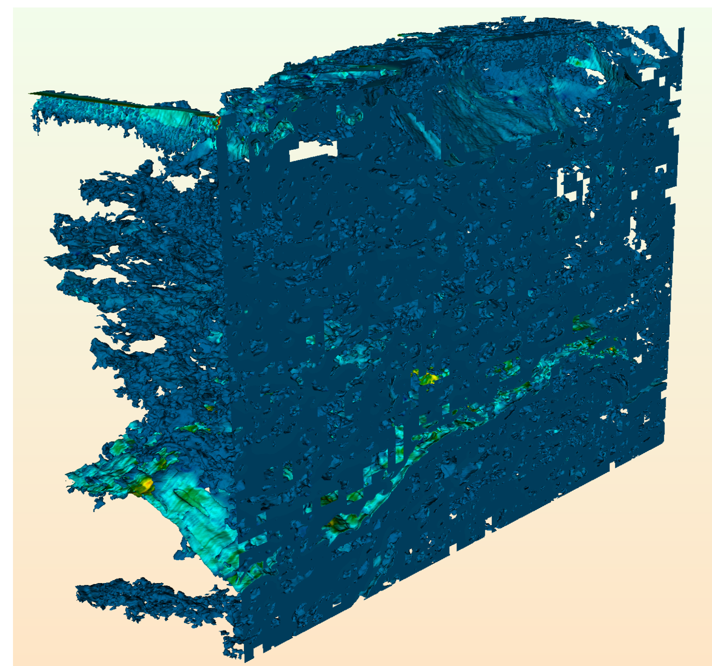
$$\frac{\partial \mathbf{u}}{\partial t} + \left(\mathbf{u}\cdot\nabla\right)\mathbf{u} + \nabla p = \nu\Delta\mathbf{u},\ \nabla\cdot\mathbf{u} = 0$$

$$\frac{\partial \rho c_k}{\partial t} + \nabla\cdot\rho\mathbf{u}c_k = \nabla\cdot\rho D_k\nabla c_k + \rho r_k$$

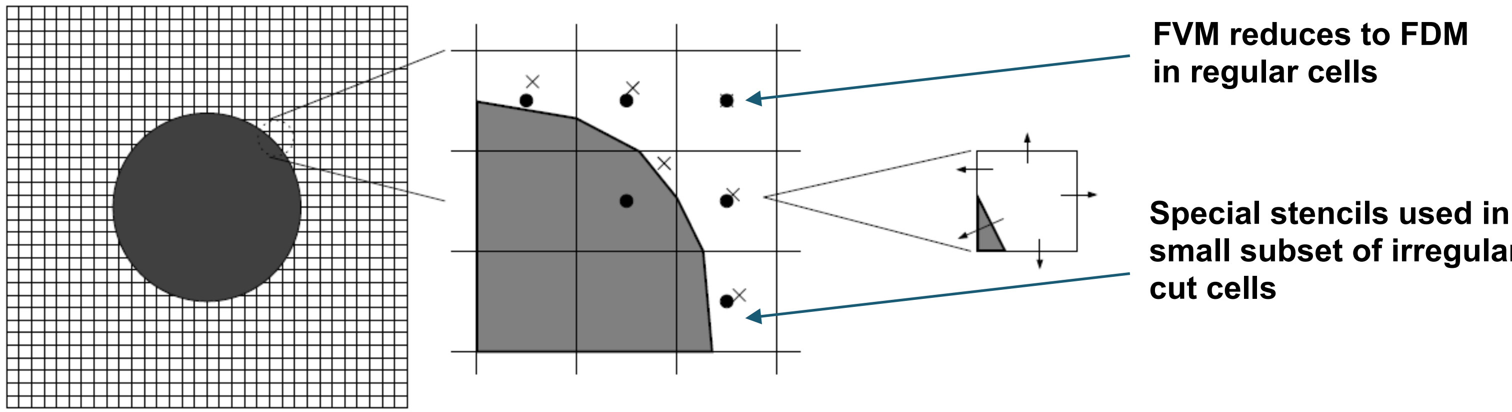
- Legacy flat MPI code
- 0.6M SLOC: C++ 90%, Fortran 10%
- Dynamic local refinement (AMR)
- 2nd-order finite-volume (low AI)
- 2nd-order Crank-Nicolson time integration or backward Euler
- PETSC AMG linear solver
- Geometry from image data
- Geochemistry: point-by-point
- Scalable (100K+ processors)



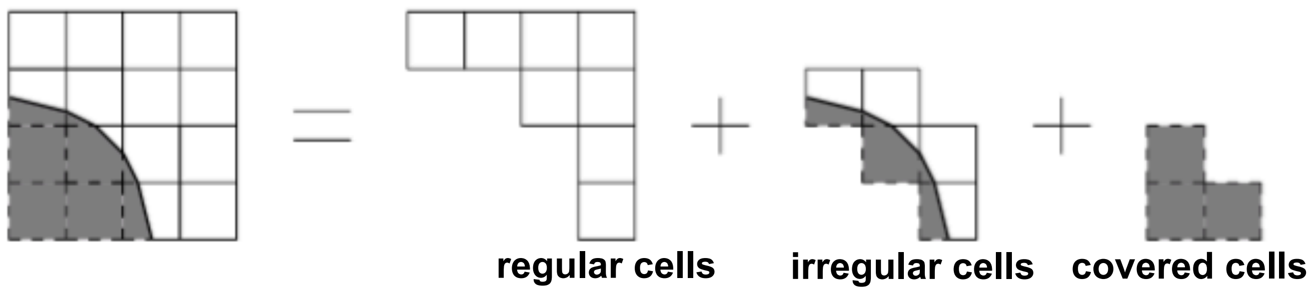




Embedded boundary method

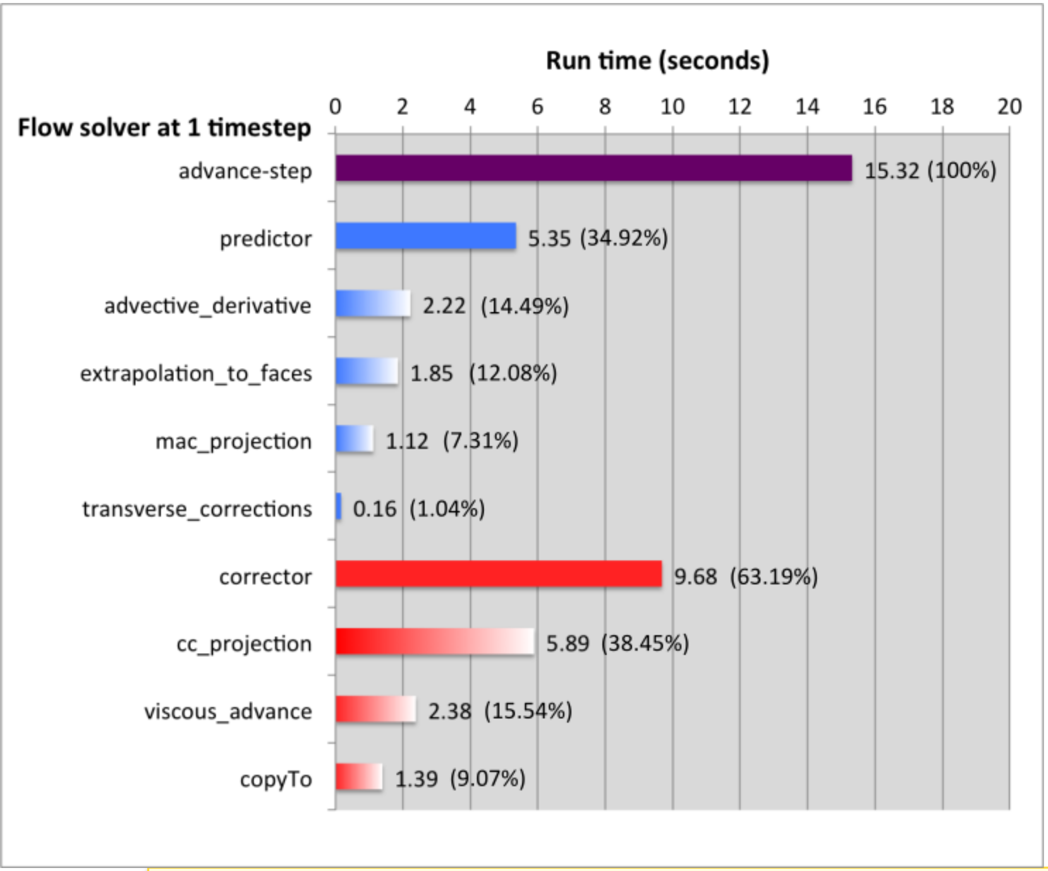


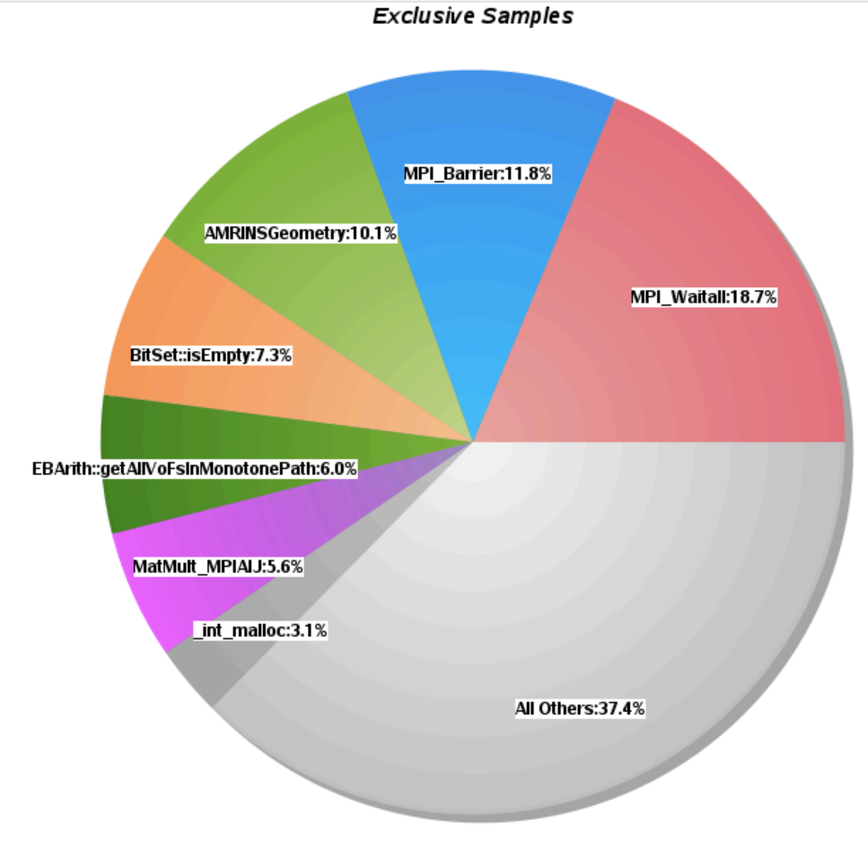
EBIndexSpace: geometric momentum (e.g. volume fractions, area fractions, centroids)
Stencils are computed at run time. Number of faces (“edges”) on irregular cells is not constant.



- Random memory access pattern (pure utilization of cache lines and vector units).
- Tiling is not straightforward.

Hotspot pattern



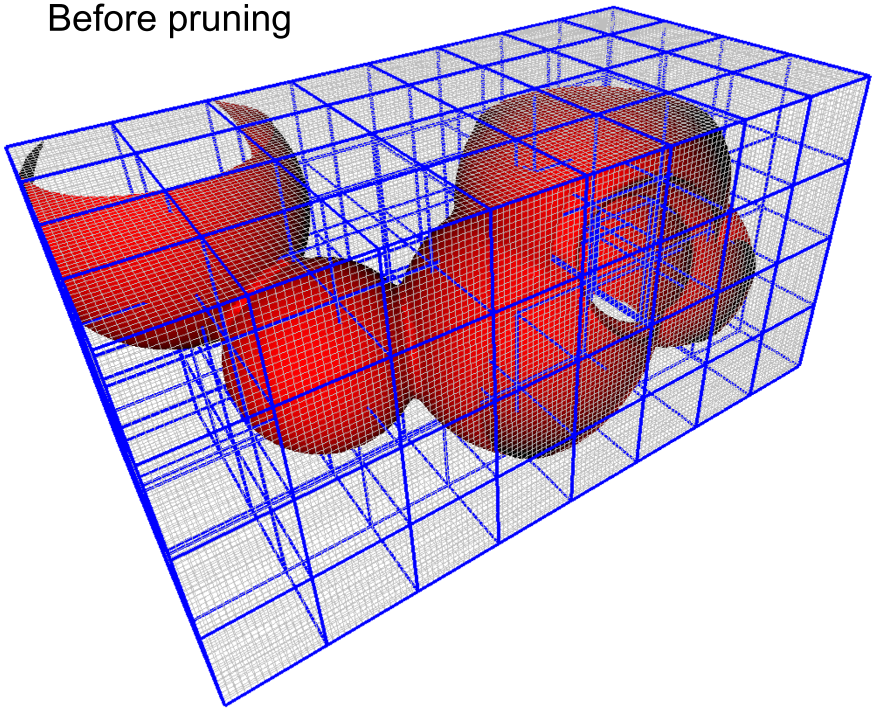


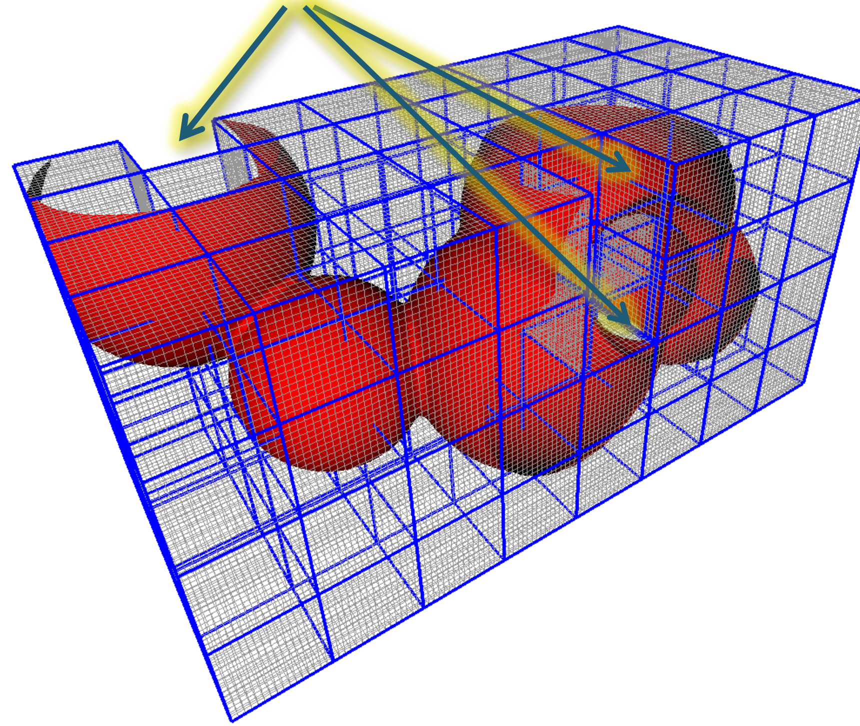
- 30-40% in Chombo: computing of fluxes, extrapolation, mass redistribution, normalization by VOF,...
- 30-40% in PETSC AMG (KSPSolve)
- 20-40% MPI (Waitall, Barrier)

Pruning of covered boxes

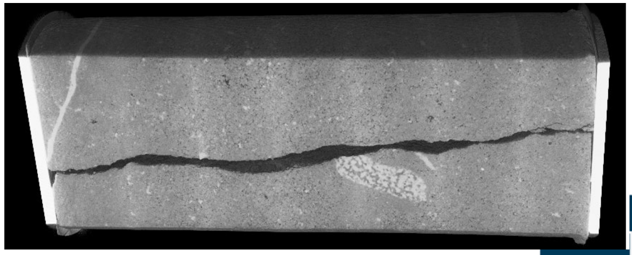
Covered boxes have been "pruned" from domain

Before pruning





Fraction of covered boxes depends on application:
it may vary **from 5% up to 70%** (e.g., fractured dolomite).
For porous media like shale it is typically ~30-40%.

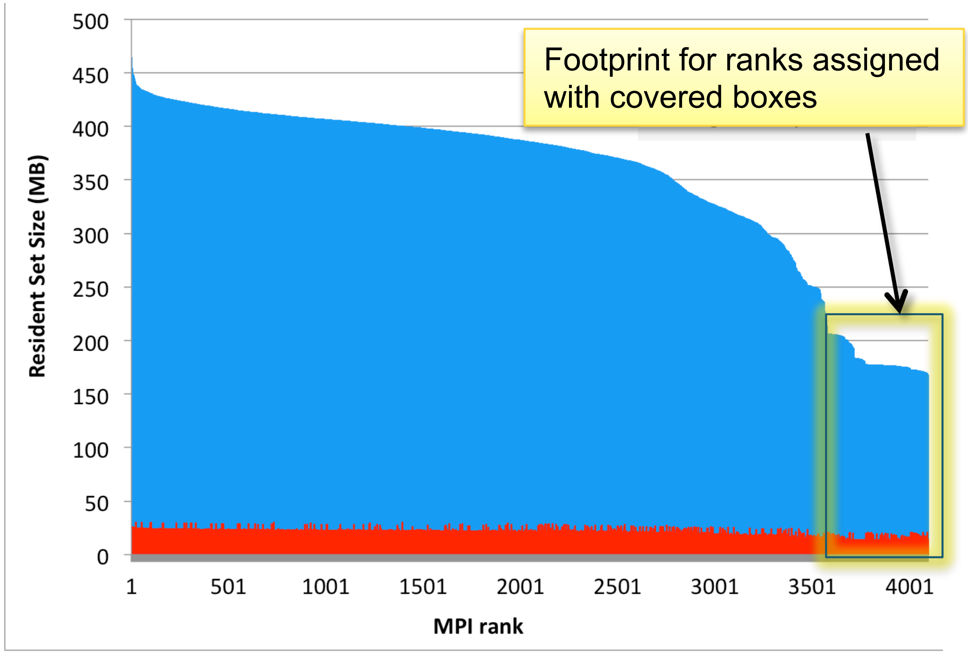


U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Box pruning: Impact on performance

Reduced memory footprint and checkpoint size



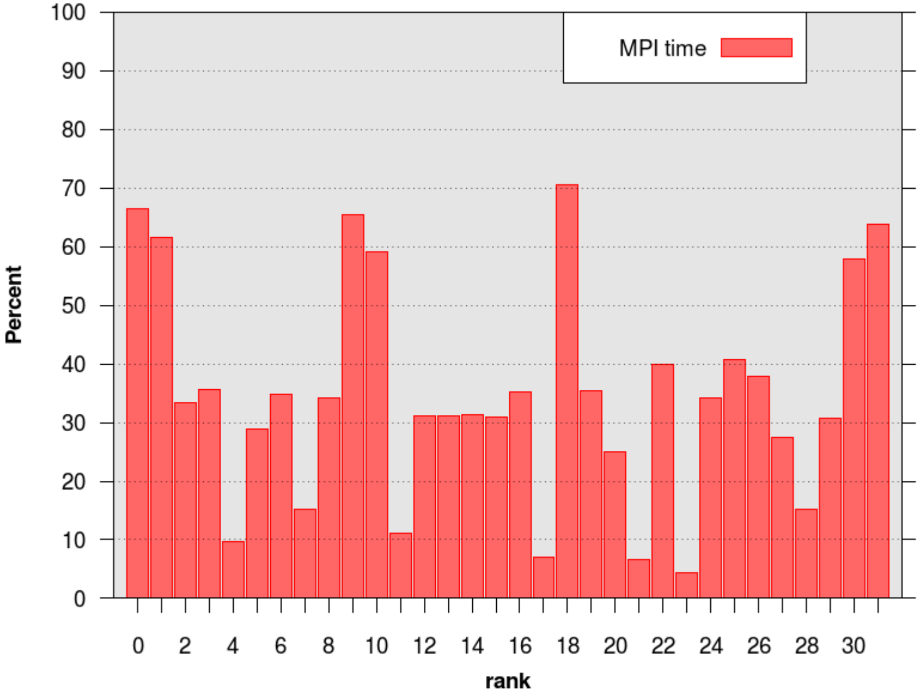
Pruning can reduce a run time as well (if the same number of PE's is used after pruning. It requires more than 1 box per PE).

# boxes	# covered boxes	plotfile [GiB]		checkpoint [GiB]		memory [GiB]	
		original	pruning	original	pruning	original	pruning
512	32 (6.25%)	7.37	6.91	6.01	5.63	216.5	210.1
4096	512 (12.5%)	59	51.62	48.09	42.08	1459	1369
32768	6862 (20.9%)	472	373.8	384.75	304.7	11211	9699

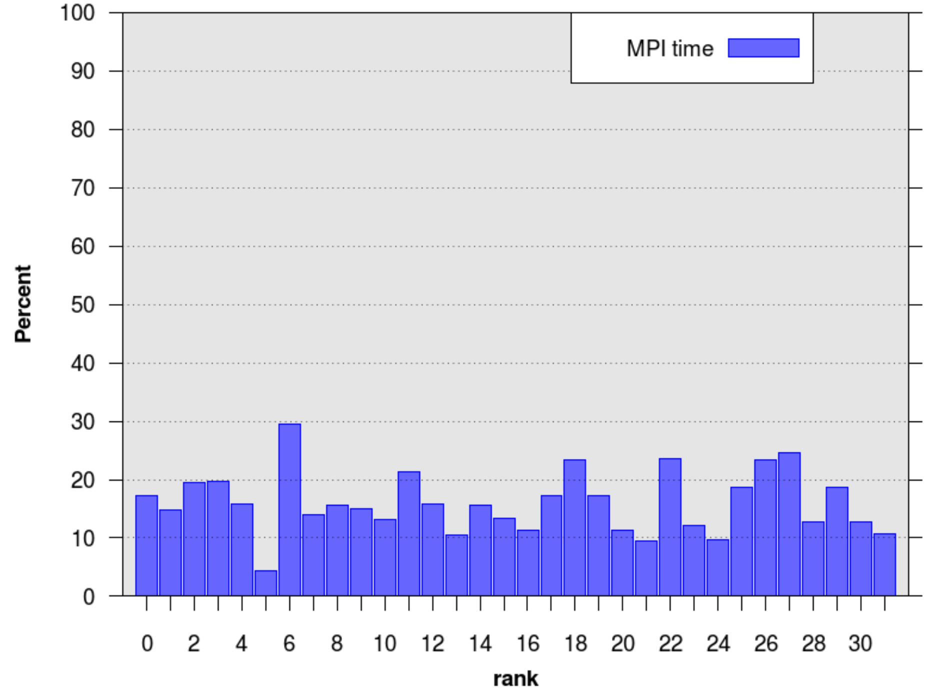
Improving of load imbalance

Communication time over MPI ranks (on single HSW node)

Load balancer based on num. of cells



Feedback-based load balancer



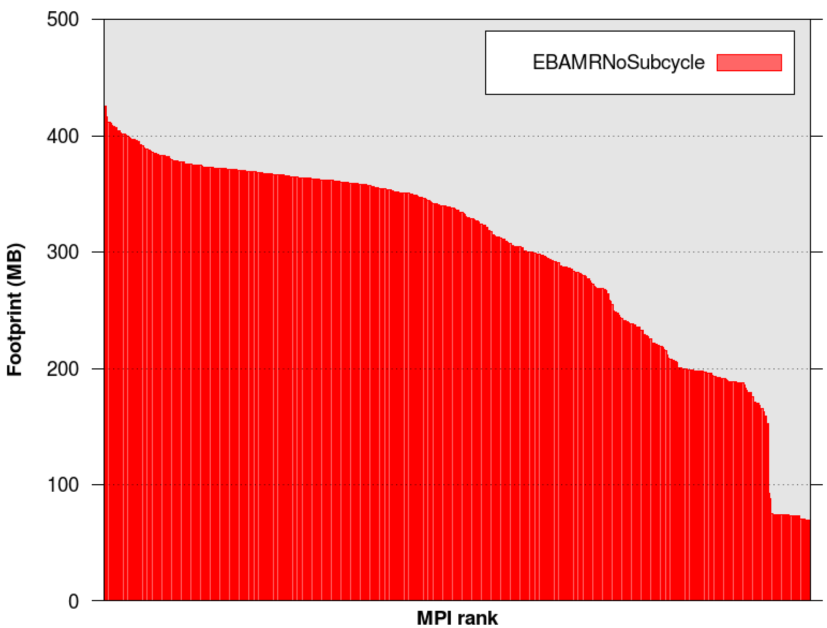
20% decrease of total run time on Cori P1.

Squeezing memory footprint

Elimination of excessive temporary variables (e.g., conservative variables) in operator for computing advective terms:
`EBAdvectLevelIntegrator` class replaced `EBPatchAdvect`
`EBAdvectPatchIntegrator` class replaced `EBLevelAdvect`

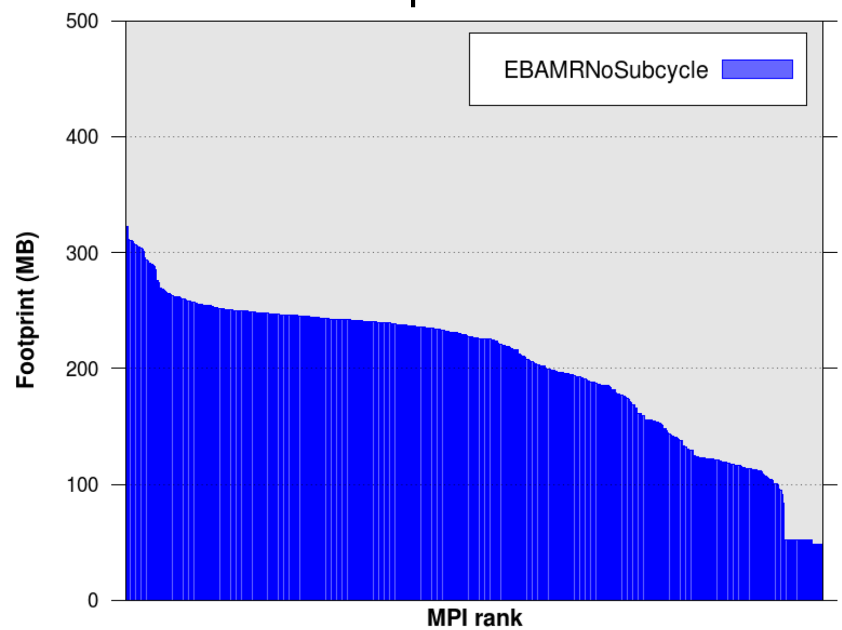
Memory footprint: 1.5x decrease

original



MPI rank: 0 to 511

optimized



MPI rank: 0 to 511

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Memory footprint: Valgrind

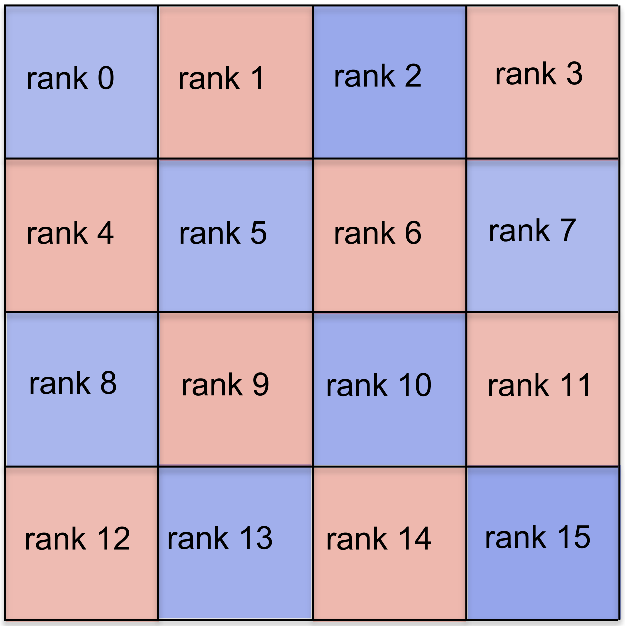
Valgrind (massif tool) provides detailed output on heap usage

```
valgrind --tool=massif ./chombo.ex
ms_print massif.out > heapDump.txt
```

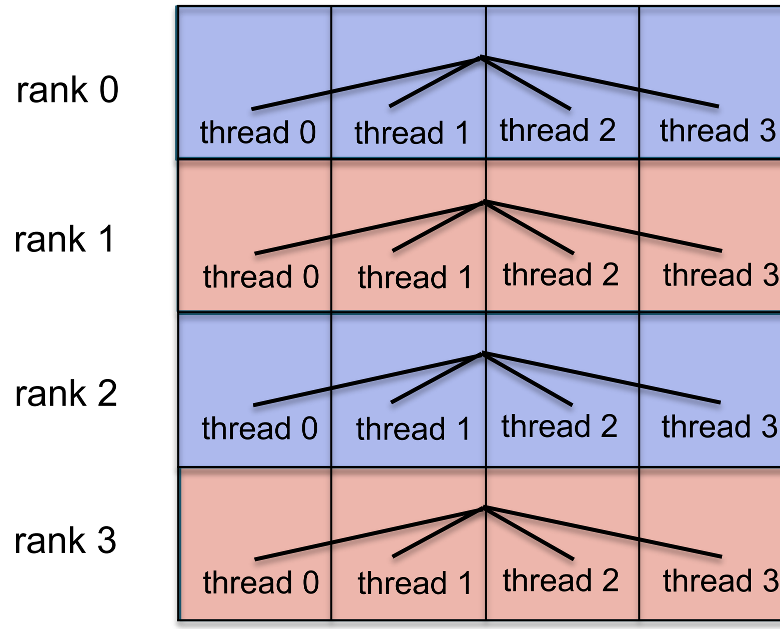
```
| ->10.99% (12,007,104B) 0x7C36EC: BaseEBFaceFAB<double>::define(EBISBox const&, Box const&, int, int) (in viscoelasticDriver3d.Linux.64.g++.gfortran.OPTHIGH.PETSC.ex)
| | | ->05.94% (6,488,064B) 0x7C15A7: EBFaceFAB::define(EBISBox const&, Box const&, int, int) (in viscoelasticDriver3d.Linux.64.g++.gfortran.OPTHIGH.PETSC.ex)
| | | | ->04.45% (4,866,048B) 0x5F4E15: EBPatchGodunov::setValidBox(Box const&, EBISBox const&, IntVectSet const&, double const&, double const&) (in viscoelasticDriver3d.Linux.64.g++.gfortran.OPTHIGH.PETSC.ex)
| | | | | ->04.45% (4,866,048B) 0x5C30C4: EBPatchAdvect::setValidBox(Box const&, EBISBox const&, IntVectSet const&, double const&, double const&)

| ->31.18% (34,078,720B) 0x47303C: BaseFab<double*>::resize(Box const&, int, double**) (in viscoelasticDriver3d.Linux.64.g++.gfortran.OPTHIGH.PETSC.ex)
| | ->31.18% (34,078,720B) 0x4A9205: BaseIVFAB<double>::define(IntVectSet const&, EBGraph const&, int const&) (in viscoelasticDriver3d.Linux.64.g++.gfortran.OPTHIGH.PETSC.ex)
| |   ->04.32% (4,718,592B) 0x5F4E95: EBPatchGodunov::setValidBox(Box const&, EBISBox const&, IntVectSet const&, double const&, double const&) (in viscoelasticDriver3d.Linux.64.g++.gfortran.OPTHIGH.PETSC.ex)
```

From flat MPI to MPI+OpenMP

16 Boxes, 16 MPI ranks



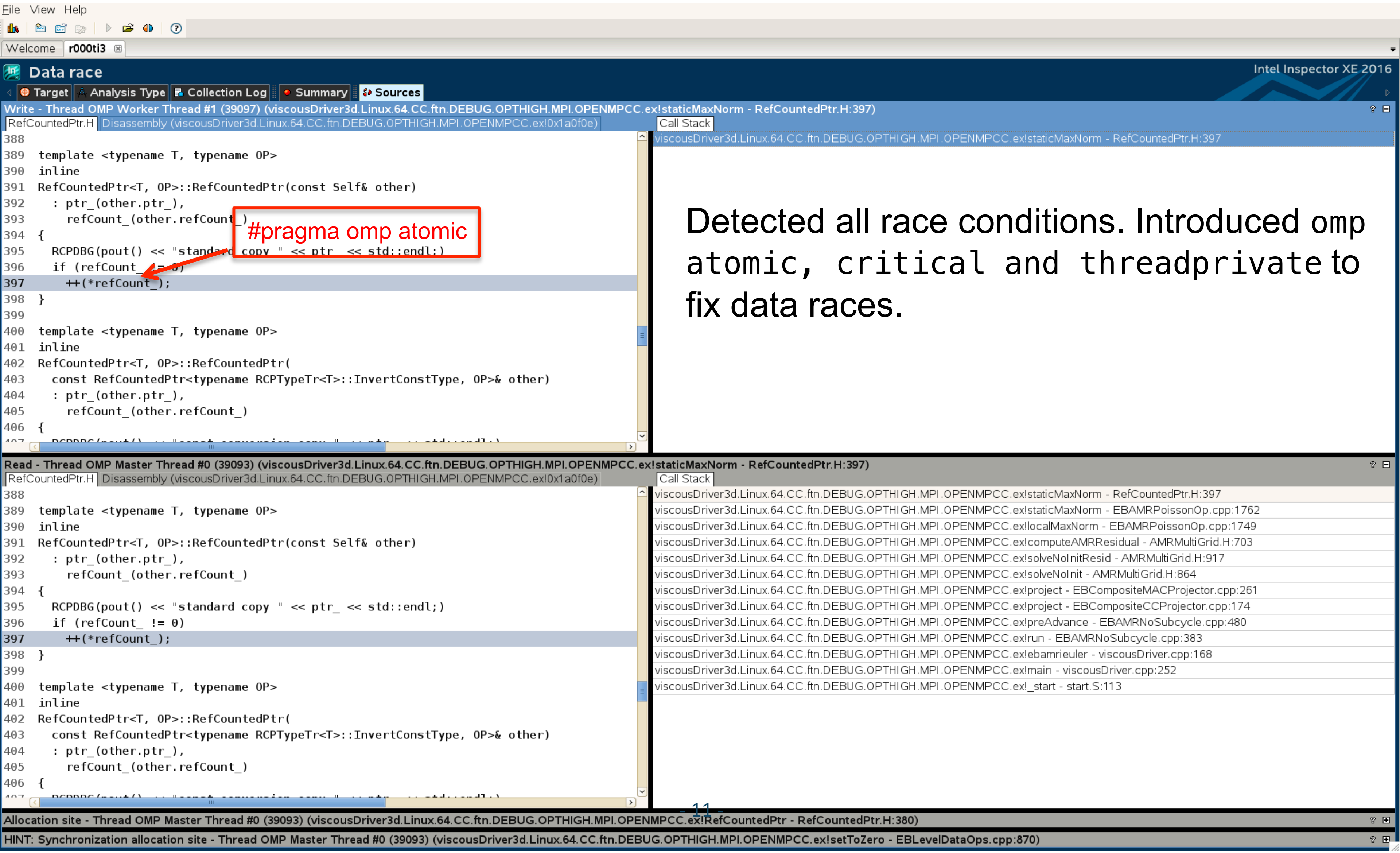
16 Boxes, 4 MPI ranks, 4 threads per rank




```
for (DataIterator dit = a_grids.dataIterator(); dit.ok(); ++dit)
  {
     averageCellToFace(a_fluxData[dit()][idir],
                       a_cellData[dit()],
                       a_grids[dit()],
                       a_ebisl[dit()],...);
  }
```

```
DataIterator dit = a_grids.dataIterator();
int nbox = dit.size();
#pragma omp parallel for
for(int mybox = 0; mybox < nbox; mybox++)
  {
     averageCellToFace(a_fluxData[dit[mybox]]
[idir],                  a_cellData[dit[mybox]],
                         a_grids[dit[mybox]],...);
  }
```

Fixing data races with Intel Inspector



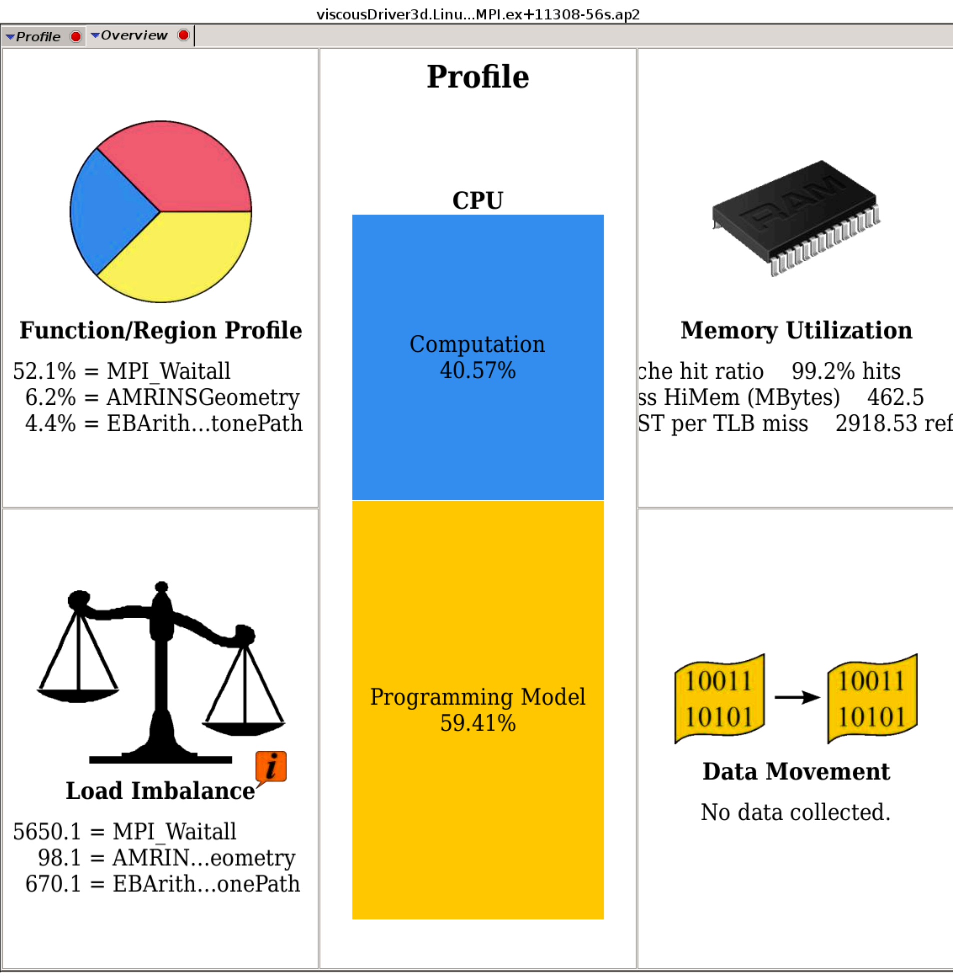
Detected all race conditions. Introduced `omp atomic`, `critical` and `threadprivate` to fix data races.

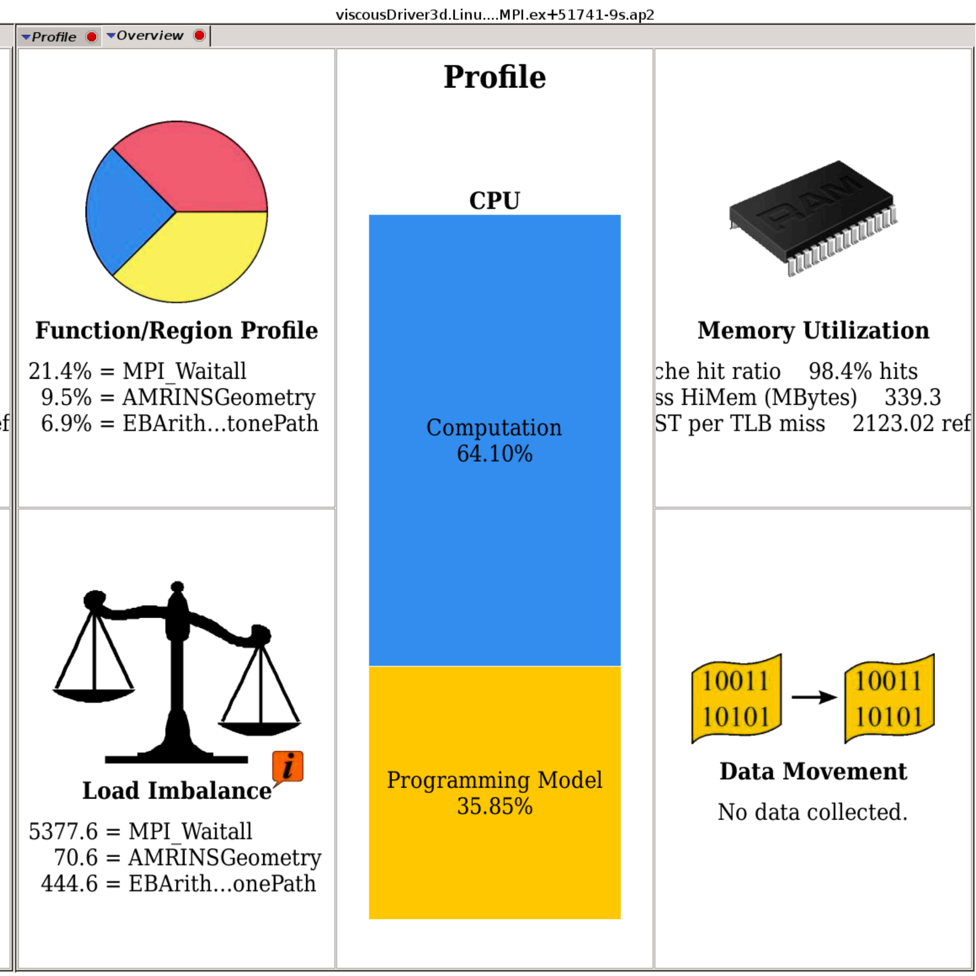
Further footprint reduction due to shared memory

1.7x less memory for MPI+OpenMP: for 6 spheres benchmark memory drops from 2880 MB/node to 1672 Mbytes/node.

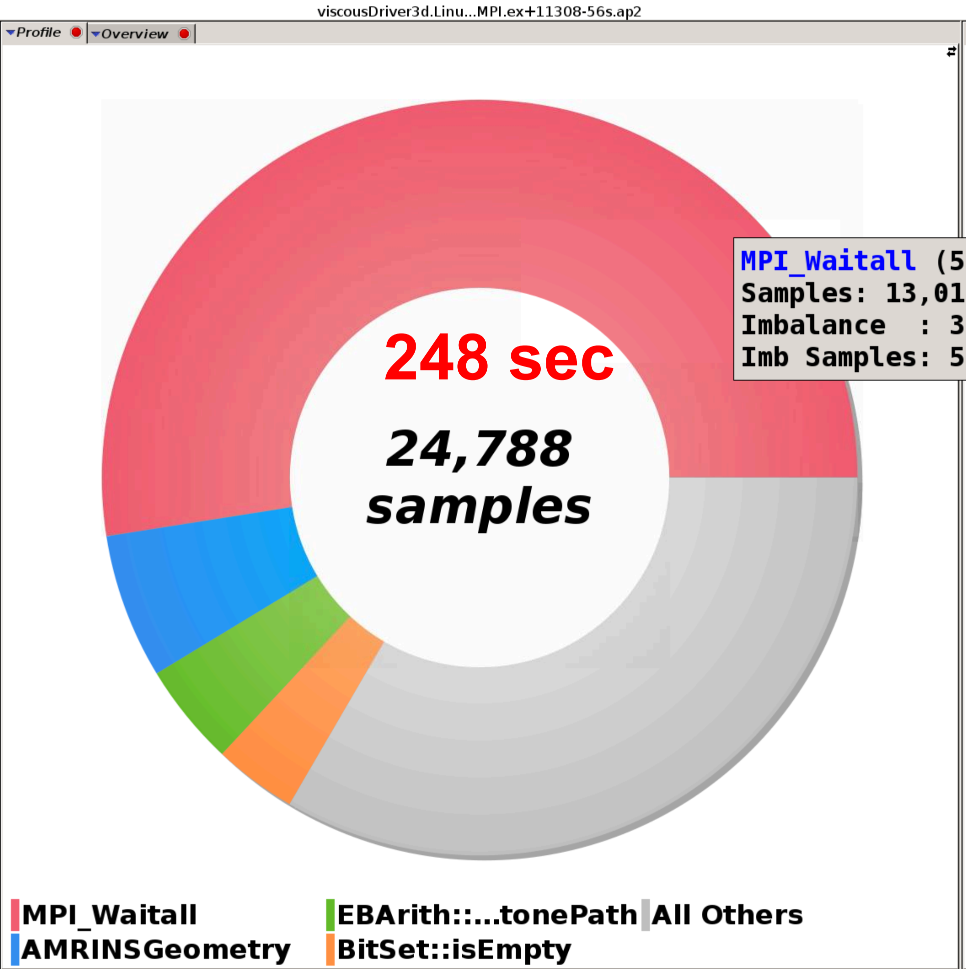
Performance of MPI+OpenMP has not been assessed. Issues with dead locks.

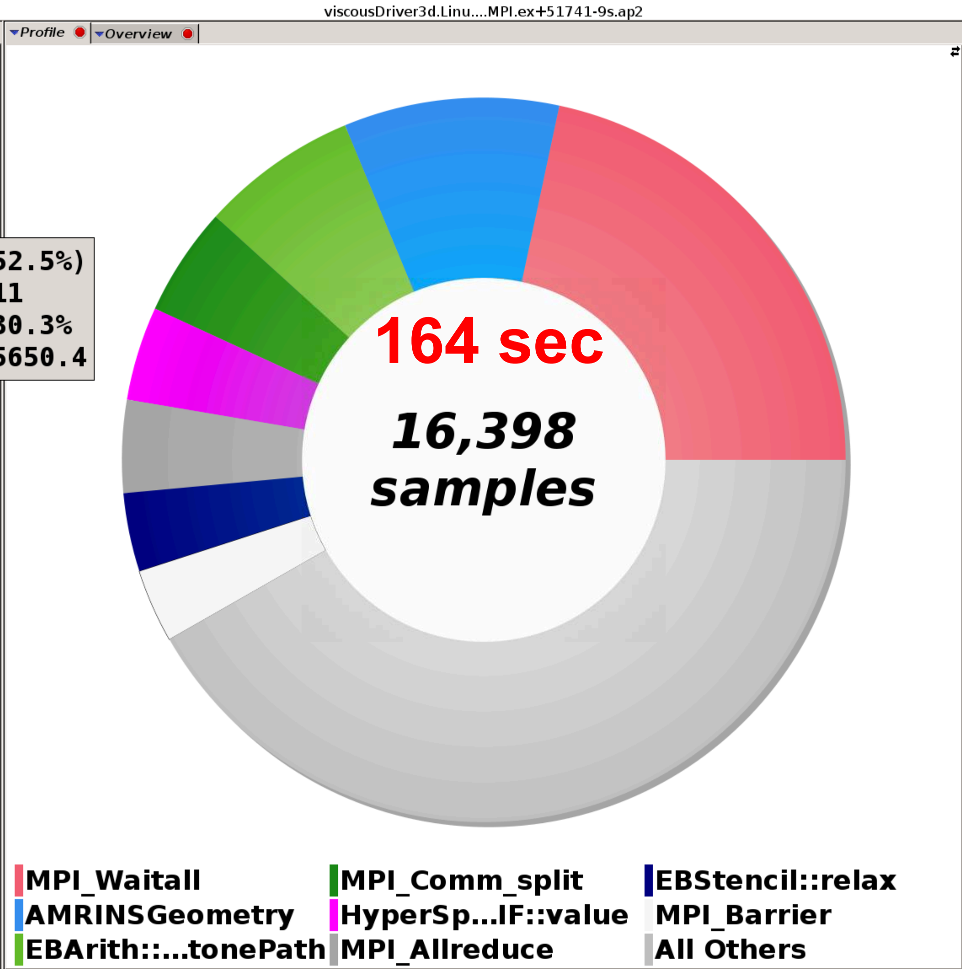
Overall performance speedup





Overall performance speedup





KNL: Single node performance (1)

Relative KNL-to-HSW performance strongly depends on complexity of geometry (i.e. fraction of irregular cells):

benchmark	GFLOPS/sec (HSW)	GFLOPS/sec (KNL)	KNL-to-HSW
100% regular (AMRPoisson)	20.7	30.8	1.49
1 sphere (0.5% irregular cells)	12.8	10.5	0.82
6 spheres (5% irregular cells)	7.5	4.4	0.58
200 spheres (15% irregular cells)	4.0	1.6	0.40

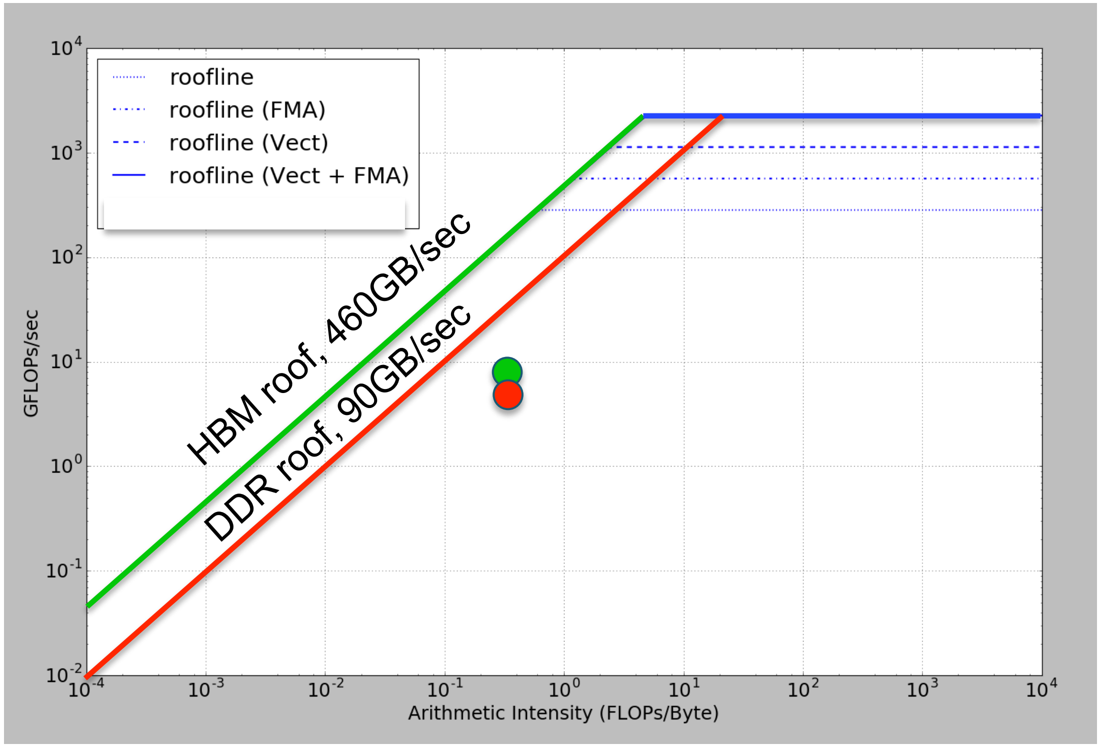
KNL: Single node performance (2)

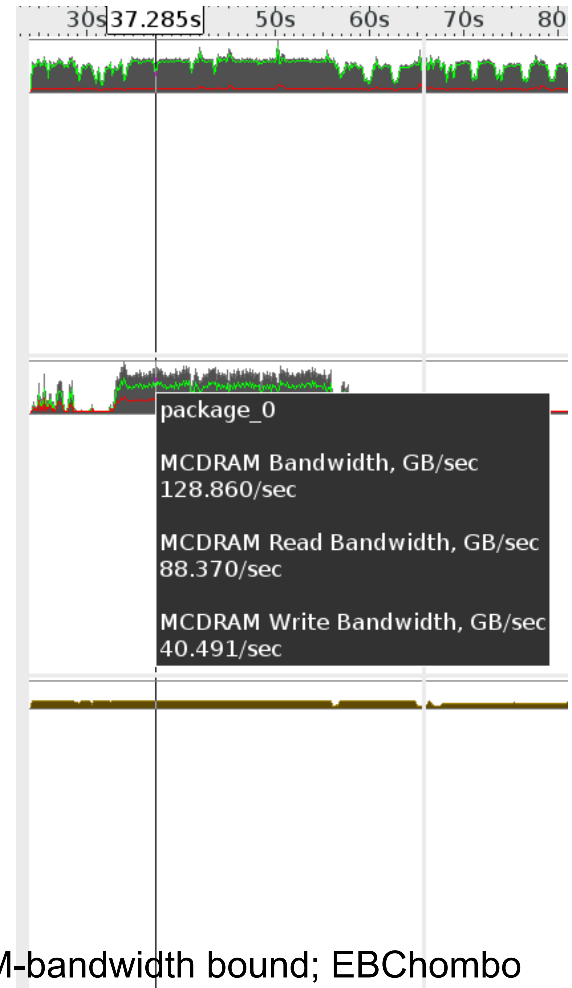
Relative KNL-to-HSW performance strongly depends on problem size:

Mesh size	GFLOPS/sec (HSW)	GFLOPS/sec (KNL, HBM)	GFLOPS/sec (KNL, DDR)
64^3	7.5	1.6	1.4
128^3	20.7	7.3	6.3
256^3	17.1	19.4	10.2
512^3	17.8	30.8	10.3

KNL: Single node performance (3)

KNL, QuadFlat (benchmark fits to HBM)



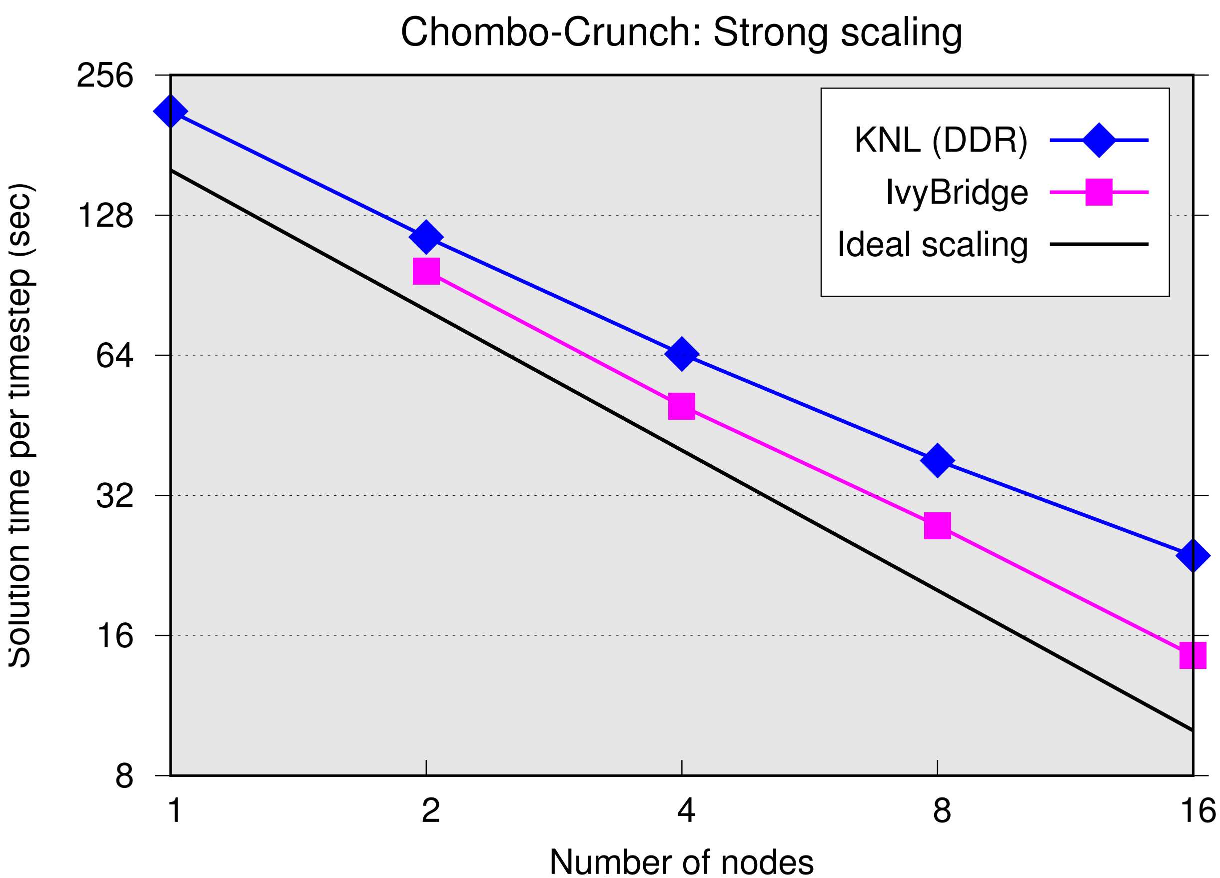


Only ~1.5x speedup using HBM vs DDR!

Kernels have different performance limitation: PETSc AMG is DRAM-bandwidth bound; EBChombo kernels (e.g., AggStencil::apply) are latency bound. Detailed roofline assessment of all major hotspots is in progress.

KNL: Multi node strong scaling study

KNL: 1 to 16 nodes on Gerty (64 MPI ranks per node, 4 remaining cores for OS)
Ivy Bridge: 1 to 16 nodes on Edison (24 MPI ranks per node)



Summary

- Reduced commun. time (MPI_Waitall) → 20-30% speedup.
- BetterEB (AggStencil) → leads to another 20% speedup.
- 1.5x reduction of memory footprint due to optimization of advection terms. 1.7x reduction due to threading.
- MPI+OpenMP development: fixed race conditions

  AMG solver (PETSC) remains unthreaded.
  - Short-term: 16 MPI ranks + 4 threads in Chombo;
    16 MPI ranks for PETSc.
  - Long-term: MPI Communication Endpoints.
- KNL-to-Haswell (single node) performance: strongly depends on geometry (fraction of irregular cells) and problem size: from 1.5x to 0.2x.
- Vectorization is pure: only 5-10% of speedup by using AVX512 vector instructions.
- Started to work on the kernel (AggStencil) to improve data locality in irregular part of computation.

Thank you.
Questions?