Adapting Minisweep, a Proxy Application, on Heterogeneous Systems Using OpenACC Directives

Sunita Chandrasekaran, Assistant Professor

Robert Searles, Ph.D. Student

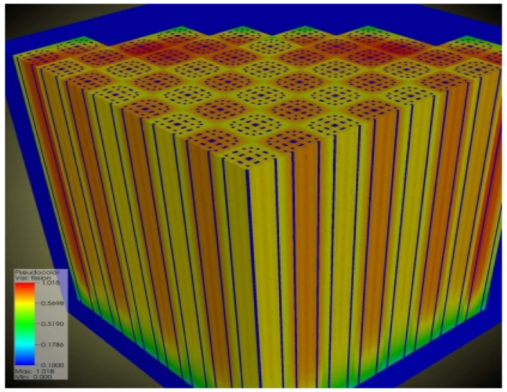
University of Delaware, DE, USA

March 27, GTC 2018, Room 211B

schandra@udel.edu

In Collaboration with Oscar Hernandez, Wayne Joubert from ORNL

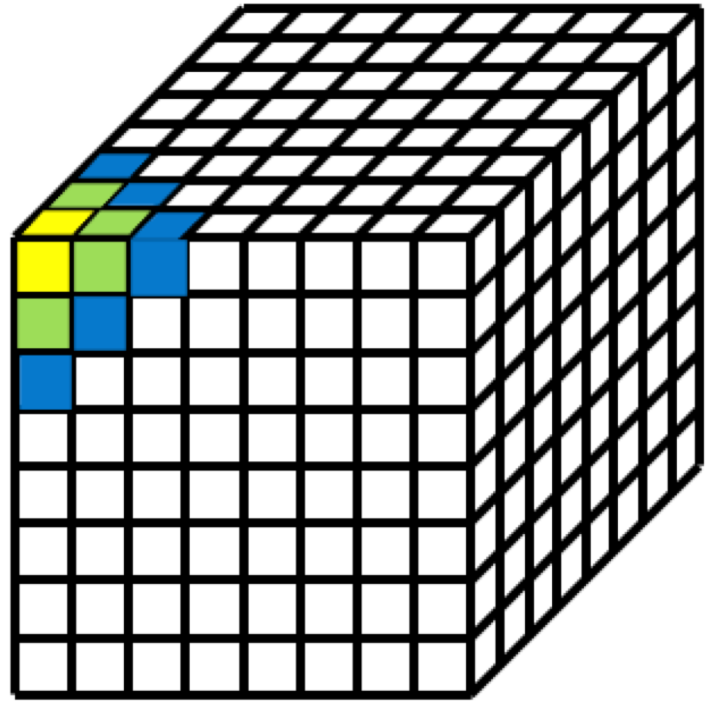
Application Overview



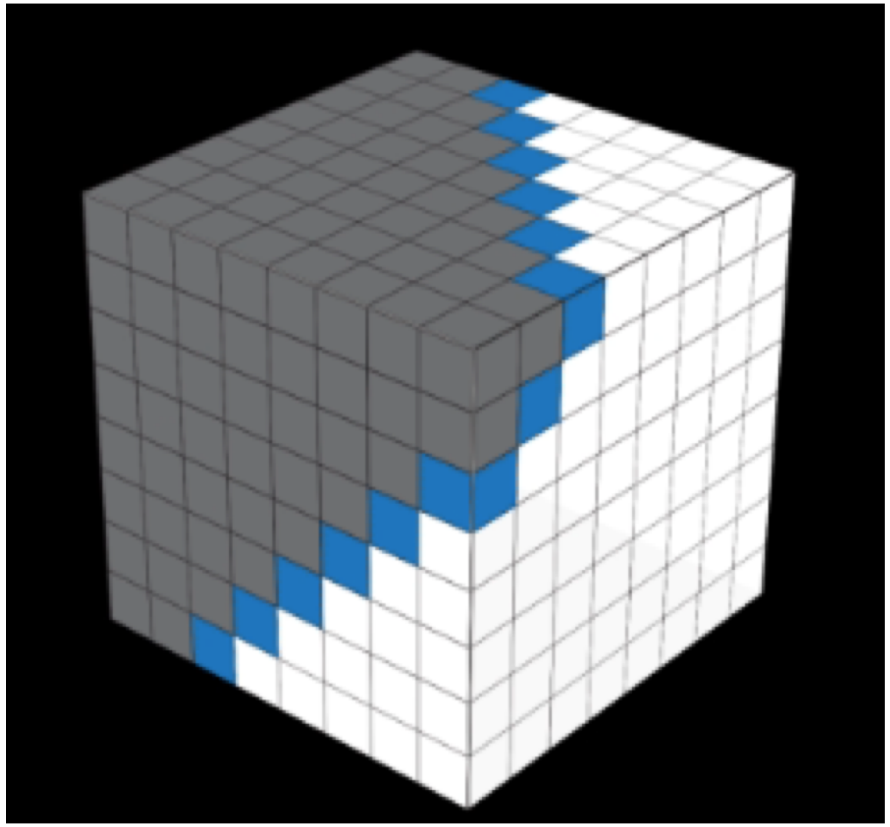
- Minisweep, a miniapp, represents 80-90% of Denovo S_n code
- Denovo S_n (discrete ordinate), part of DOE INCITE project, is used to model fusion reactor – CASL, ITER
  - **Impact:** By running Minisweep faster, experiments with more configurations can be performed directly impacting the determination of accuracy of radiation shielding
  - **Impact:** High-fidelity predictive capability for component and system performance
- Poses a six dimensional problem
  - 3D in space, 2D in angular particle direction and 1D in particle energy
- The parallel pattern observed is wavefront-based

Overview of Sweep Algorithm

- 5 nested loops
  - X, Y, Z dimensions, (outer loops)
  - Energy Groups, Moments, Angles (in-grid cells)
  - Upstream data dependency
- Challenge to achieve high performance
  - sparse hyperbolic PDE solvers are generally limited to very low computational intensities
- Goal to expose as much as thread parallelism as possible, maximize locality of reference, reduce cost of data transfer



Parallelizing Sweep Algorithm
(Video/Image Credit – Evan Krape, UDEL)



Parallelizing Sweep Algorithm: KBA

- Koch-Baker-Alcouffe (KBA)
- Algorithm developed in 1992 at Los Alamos
- Parallel sweep algorithm that overcomes some of the dependencies using a wavefront.

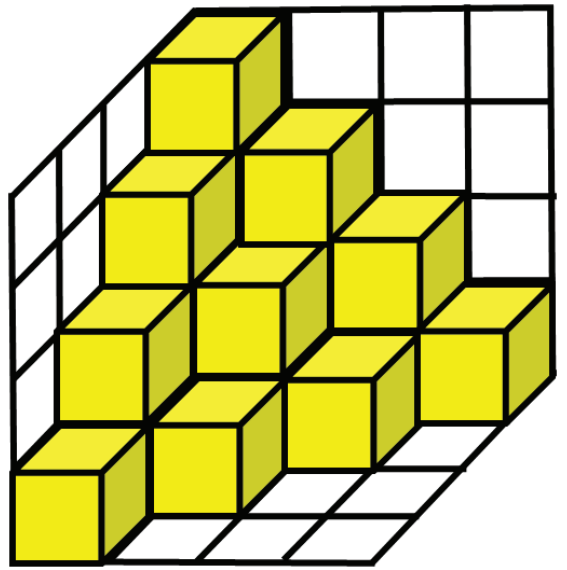


Image credit: High Performance Radiation Transport Simulations: Preparing for TITAN
C. Baker, G. Davidson, T. M. Evans, S. Hamilton, J. Jarrell and W. Joubert
ORNL, USA

Work somewhat similar to KBA

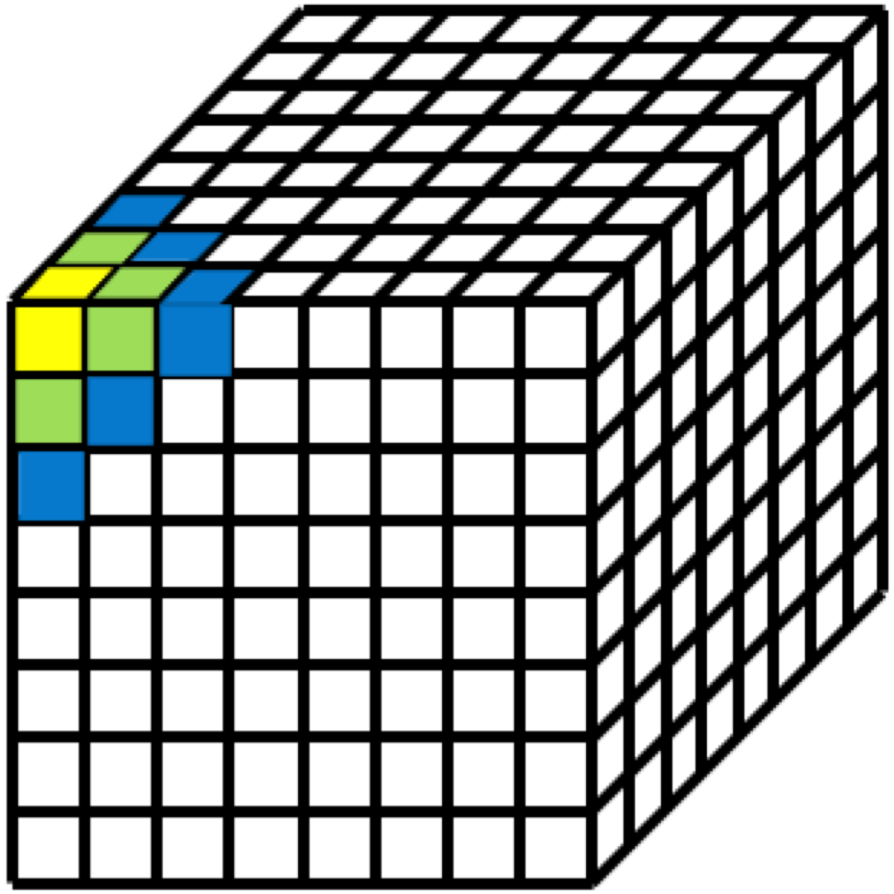
- Past work wasn't accelerating across all problem dimensions (space, octant, angle, moment, energy groups)
- Sweep3D – Los Alamos National Laboratory (LANL) on GPUs
- Algorithm parallelized in space and vectorized in angle for the IBM Cell processor

Expressing wavefront via software abstractions – A Challenge

- Existing solutions involve manual rewrites, or compiler-based loop transformations
  - Michael Wolfe. 1986. Loop skewing: the wavefront method revisited. *Int. J. Parallel Program.* 15, 4 (October 1986), 279-293. DOI=http://dx.doi.org/10.1007/BF01407876
- CHiLL framework, a polyhedral compiler for transformation framework
- No solution in high-level languages like OpenMP/OpenACC
- No software abstractions

Using OpenACC to program KBA

- Spatial decomposition = outer layer (KBA)
  - No existing abstraction for this
- In-gridcell computations = inner layer
  - Application specific
- Upstream data dependencies
  - Slight variation between wavefront applications



Using OpenACC to program KBA

- Storing all previous wavefronts is unnecessary
  - How many neighbors and prior wavefronts are accessed?
- Face arrays make indexing easy
  - Smaller data footprint
- Limiting memory to the size of the largest wavefront is optimal, but not practical

```
/*--- Loop over wavefronts ---*/
for (wavefront = 0; wavefront < num_wavefronts; wavefront+=1) {

   /*---KBA threading---*/
#pragma acc loop independent gang, collapse(2)
    for( iy=0; iy<dim_y; ++iy )
    for( ix=0; ix<dim_x; ++ix ) {

      int iz = wavefront - (ix + iy);
      if (iz >= 0 && iz <= wavefront && iz < dim_z) {

            /*---moments to angles---*/
#pragma acc loop independent vector, collapse(3)
        for( ie=0; ie<dim_ne; ++ie )
            for( iu=0; iu<NU; ++iu )
            for( ia=0; ia<dim_na; ++ia ) {
               P result = (P)0;
#pragma acc loop seq
               for( im=0; im < dim_nm; ++im )
                { /*---moments to angles conversion---*/ }
            }

        /*---solve---*/
        #pragma acc loop independent vector, collapse(2)
        for( ie=0; ie<dim_ne; ++ie )
            for( ia=0; ia<dim_na; ++ia )
             { /*---solve calculation---*/ }

            /*---angles to moments---*/
#pragma acc loop independent vector, collapse(3)
        for( ie=0; ie<dim_ne; ++ie )
            for( iu=0; iu<NU; ++iu )
            for( im=0; im<dim_nm; ++im ) {
#pragma acc loop seq
               P result = (P)0;
               for( ia=0; ia<dim_na; ++ia )
                { /*---angles to moments conversion---*/ }
        }
      }
    }
  }
```

Create software abstractions for Wavefront

- Avoid manual loop restructuring
- Analyze flow of data and computation in wavefront codes
- Memory model abstraction
- Wavefront loop transformation algorithm
- Need to address multiple layers of parallelism (minisweep 5-levels)

Minisweep code status

- Github: https://github.com/wdj/minisweep
- Was ported to CUDA and OpenMP targeting Beacon and TITAN at ORNL
- Being currently used for SummitDev and Summit acceptance testing at ORNL

Experimental Setup

- NVIDIA PSG Cluster
  - CPU: Intel Xeon E5-2698 v3 (16-core)
  - GPU: NVIDIA Tesla P100, Tesla V100, and Tesla K40 (4 GPUs per node)

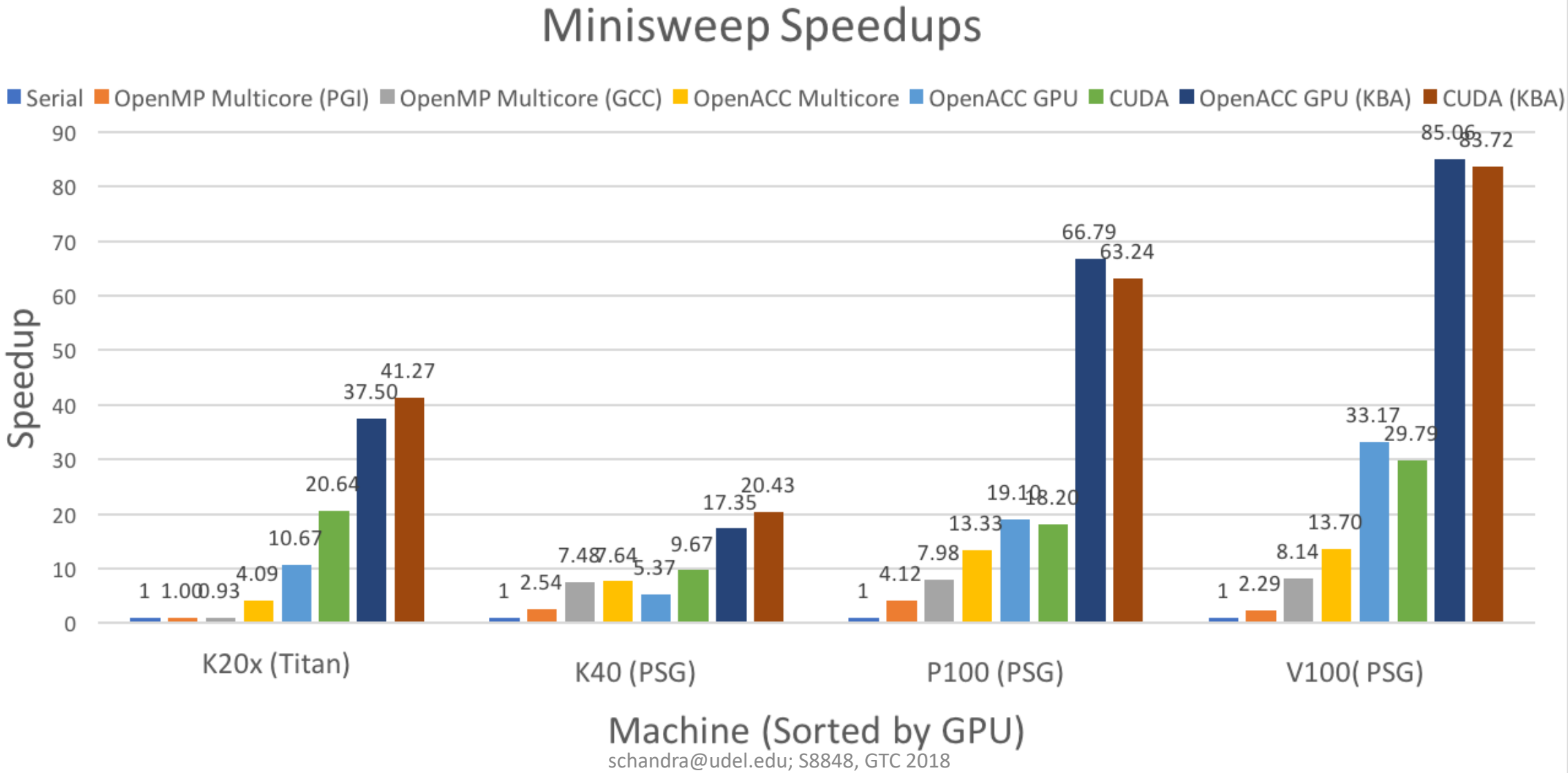
- ORNL Titan
  - CPU: AMD Opteron 6274 (16-core)
  - GPU: NVIDIA Tesla K20x

- PGI OpenACC Compiler 17.10
- OpenMP – GCC 6.2.0 (we used Intel 17.0 compiler too but GCC performed better)

Input Parameters

- Scientifically (Rep. runs within Denovo)
  - X/Y/Z dimensions = 64 (you could have diff. values for the 3 dimensions)
  - # Energy Groups = 64
  - # Angles = 32
- Goal is to explore larger spatial dimensions

Results


Minisweep Speedups
Serial
OpenMP Multicore (PGI)
OpenMP Multicore (GCC)
OpenACC Multicore
OpenACC GPU
CUDA
OpenACC GPU (KBA)
CUDA (KBA)
Speedup
90
80
70
60
50
40
30
20
10
0
1
1.00
0.93
4.09
10.67
20.64
37.50
41.27
1
2.54
7.48
7.64
5.37
9.67
17.35
20.43
1
4.12
7.98
13.33
19.10
18.20
66.79
63.24
1
2.29
8.14
13.70
33.17
29.79
85.06
83.72
K20x (Titan)
K40 (PSG)
P100 (PSG)
V100(PSG)
Machine (Sorted by GPU)

Results, On-going work

- Parallelized the in-grid cell computations,
- Also the spatial decomposition utilizing the KBA parallel sweep algorithm to resolve data dependencies
- Maintained a single code base for CPUs, GPUs
- OpenACC implementation on Volta GPU shows 85.06x over 83.72x using CUDA
- On-going work - multidirectional-sweep

Takeaway(s)

- Application of directives to a code is not magical !!
- It takes several iterative steps to get it right and get comparable speedup
- Time taken for development can vary depending on programmers' expertise
- Provide feedback to the "user-driven" OpenACC standard committee about need for directives/software abstractions - this is a KEY step for standard's progress.
- Let's talk if you have a code that demonstrates need for a directive, which currently does not exist in the standard – Contact me schandra@udel.edu

- ACK: Mat Colgrove (PGI), Pat Brooks (PGI) OpenACC Standard Committee
- Many thanks to NVIDIA for giving us access to their PSG cluster!