Abstractions and Directives for Adapting Wavefront Algorithms to Future Architectures

Robert Searles, Sunita Chandrasekaran
(rsearles, schandra)@udel.edu
Wayne Joubert, Oscar Hernandez
(joubert,oscar)@ornl.gov

PASC 2018 June 3, 2018

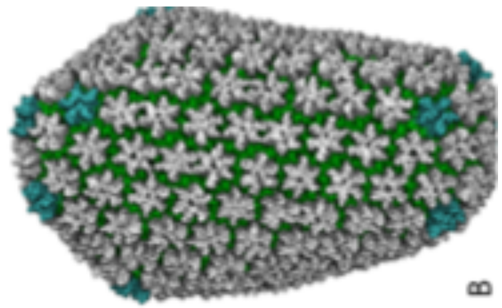
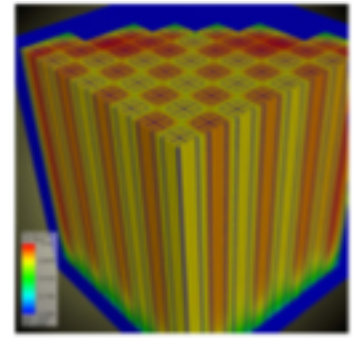
Motivation

- Parallel programming is software's future
  - Acceleration
- State-of-the-art abstractions handle simple parallel patterns well
- Complex patterns are hard!

Our Contributions

- An abstract representation for wavefront algorithms
- A performance portable proof-of-concept of this abstraction using directives: OpenACC
  - Evaluation on multiple state-of-the-art platforms
- A description of the limitations of existing high-level programming models

Several ways to accelerate

Applications

Libraries

Programming Languages

Directives

Drop in acceleration

Maximum Flexibility

Used for easier acceleration

Directive-Based Programming Models

- OpenMP (current version 4.5)
  - Multi-platform shared multiprocessing API
  - Since 2013, supporting device offloading
- OpenACC (current version 2.6)
  - Directive-based model for heterogeneous computing

OpenMP OpenACC
More Science, Less Programming

Serial Example

```
for (int i = 0; i < N; i++) {
        c[i] = a[i] + b[i];
}
```

OpenACC Example

```
#pragma acc parallel loop independent
for (int i = 0; i < N; i++) {
        c[i] = a[i] + b[i];
}
```

CUDA Example

Host Code

```
cudaError_t cudaStatus;

// Choose which GPU to run on, change this on a multi-GPU system.
cudaStatus = cudaSetDevice(0);

// Allocate GPU buffers for three vectors (two input, one output)
cudaStatus = cudaMalloc((void**)&dev_c, N* sizeof(int));
cudaStatus = cudaMalloc((void**)&dev_a, N* sizeof(int));
cudaStatus = cudaMalloc((void**)&dev_b, N* sizeof(int));

// Copy input vectors from host memory to GPU buffers.
cudaStatus = cudaMemcpy(dev_a, a, N* sizeof(int), cudaMemcpyHostToDevice);
cudaStatus = cudaMemcpy(dev_b, b, N* sizeof(int), cudaMemcpyHostToDevice);

// Launch a kernel on the GPU with one thread for each element.
addKernel<<<N/BLOCK_SIZE, BLOCK_SIZE>>>(dev_c, dev_a, dev_b);

// cudaThreadSynchronize waits for the kernel to finish, and returns
// any errors encountered during the launch.
cudaStatus = cudaThreadSynchronize();

// Copy output vector from GPU buffer to host memory.
cudaStatus = cudaMemcpy(c, dev_c, N* sizeof(int), cudaMemcpyDeviceToHost);

cudaFree(dev_c);
cudaFree(dev_a);
cudaFree(dev_b);

return cudaStatus;
```

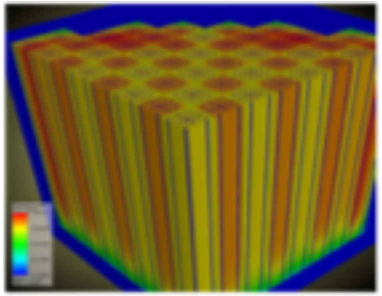
Kernel

```
__global__ void addKernel(int *c, const int *a, const int *b)
{
   int i = threadIdx.x + blockIdx.x * blockDim.x;
   c[i] = a[i] + b[i];
}
```

Pattern-Based Approach in Parallel Computing

- Several parallel patterns
  - Existing high-level languages provide abstractions for many simple patterns
- However there are complex patterns often found in scientific applications that are a challenge to be represented with software abstractions
  - Require manual code rewrite
- Need additional features/extensions!
  - How do we approach this? (Our paper's contribution)

Application Motivation: Minisweep

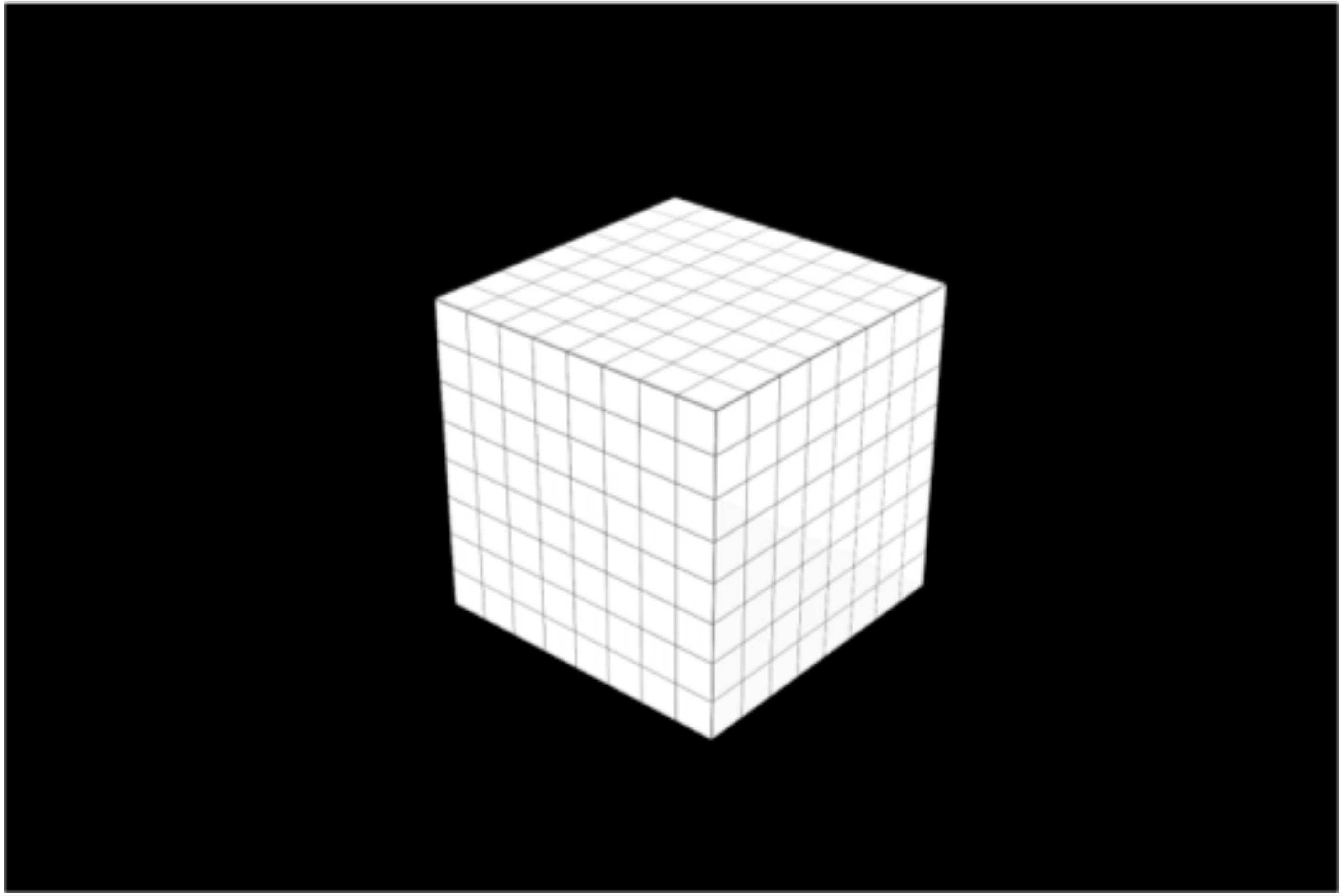


- A miniapp modeling wavefront sweep component of Denovo Radiation transport code from ORNL
  - Minisweep, a miniapp, represents 80-90% of Denovo
- Denovo - part of DOE INCITE project, is used to model fusion reactor – CASL, ITER
- Run many times with different parameters
- The faster it runs, the more configurations we can explore
- Poses a six dimensional problem
- 3D in space, 2D in angular particle direction and 1D in particle energy

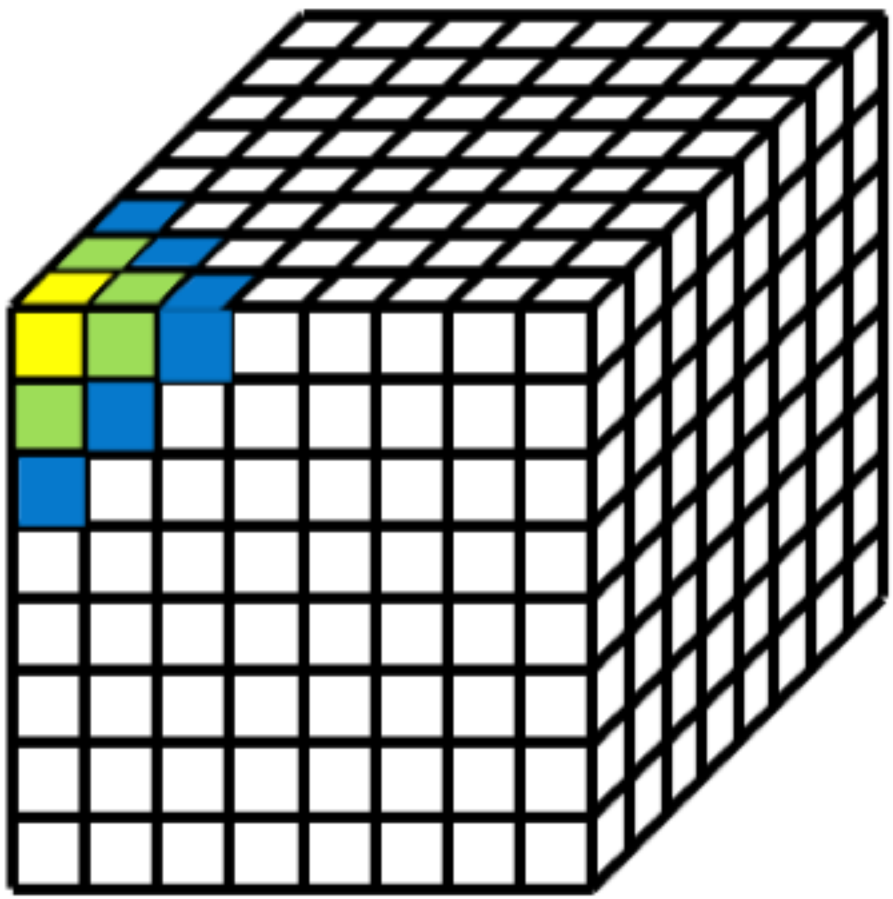
Minisweep code status

- Github: https://github.com/wdj/minisweep
- Early application readiness on ORNL Titan
- Being used for #1 TOP500 -> Summit acceptance testing
- Has been ported to Beacon and Titan (ORNL machines) using OpenMP and CUDA

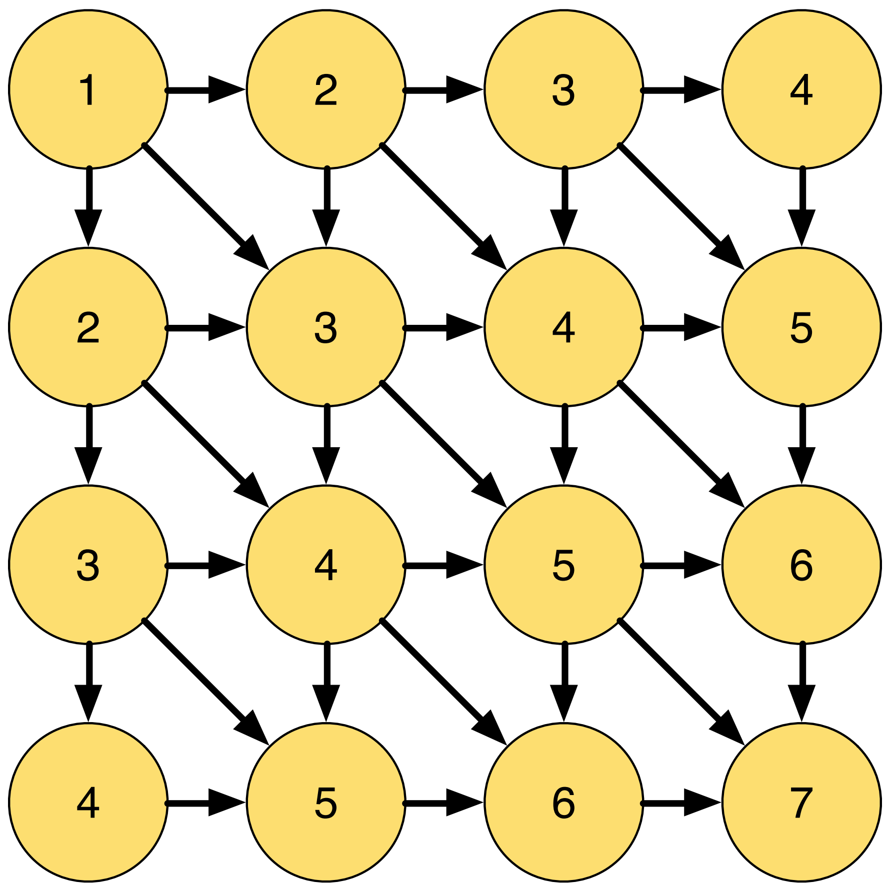
Minisweep: The Basics



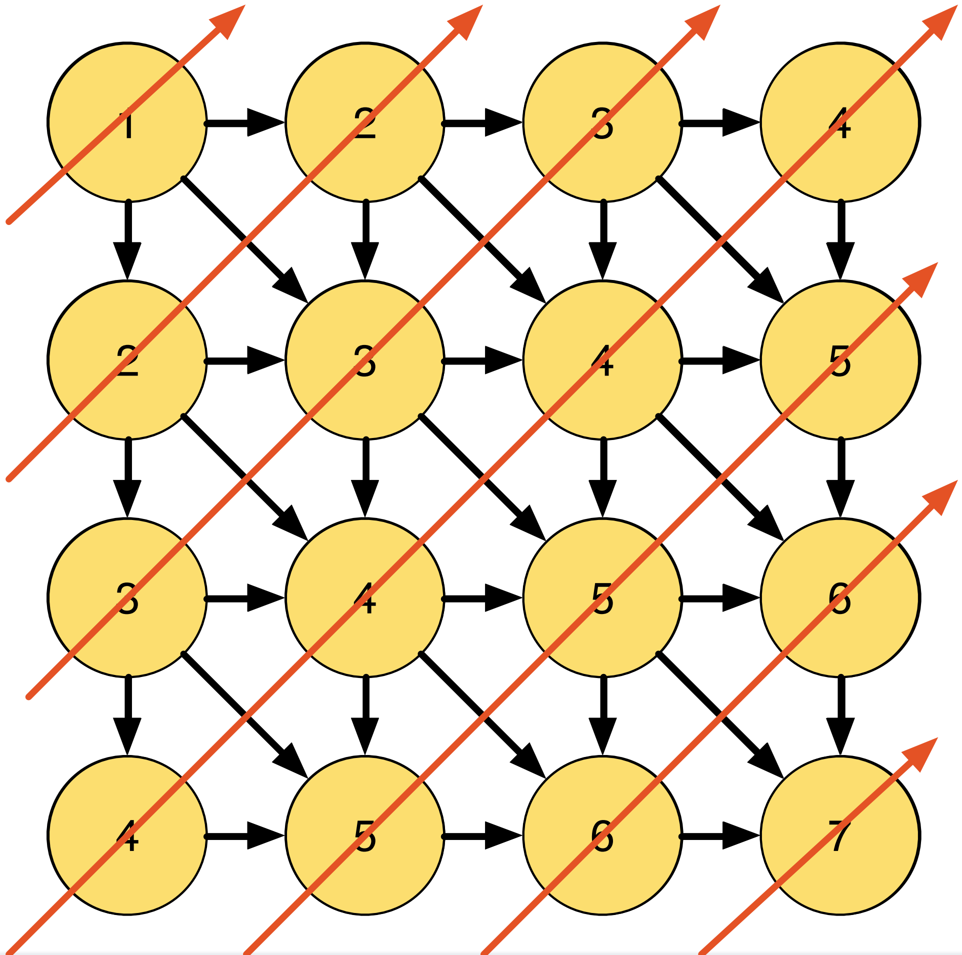
Parallelizing Sweep Algorithm



Complex Parallel Pattern Identified: Wavefront



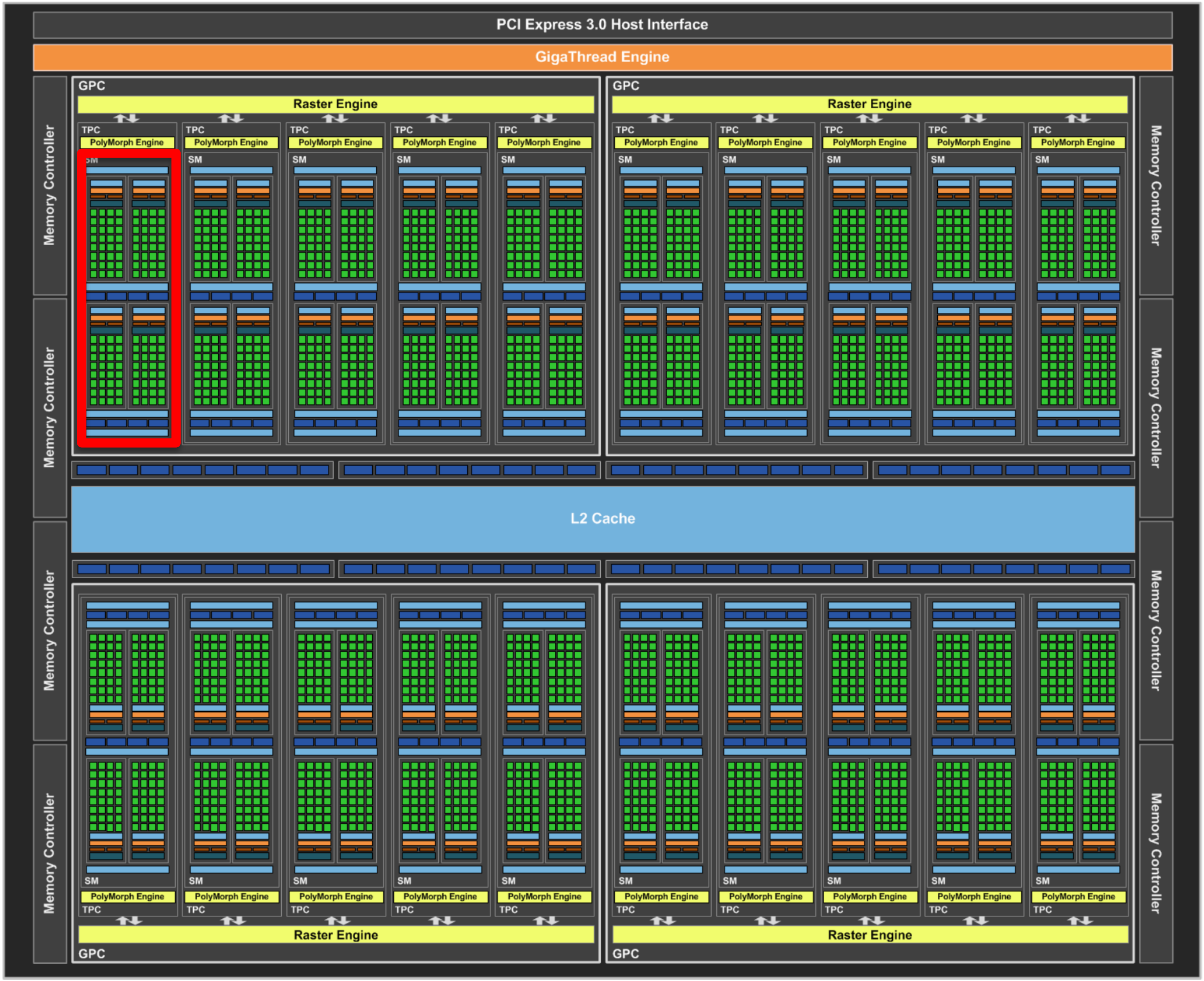
Complex Parallel Pattern Identified: Wavefront



Overview of Sweep Algorithm

- 5 nested loops
  - X, Y, Z dimensions, Energy Groups, Angles
  - OpenACC/PGI only offers 2 levels of parallelism: gang and vector (worker clause not working properly)
  - Upstream data dependency

PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
TPC
PolyMorph Engine
SM
Memory Controller
L2 Cache


PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
TPC
PolyMorph Engine
SM
Memory Controller
L2 Cache

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.

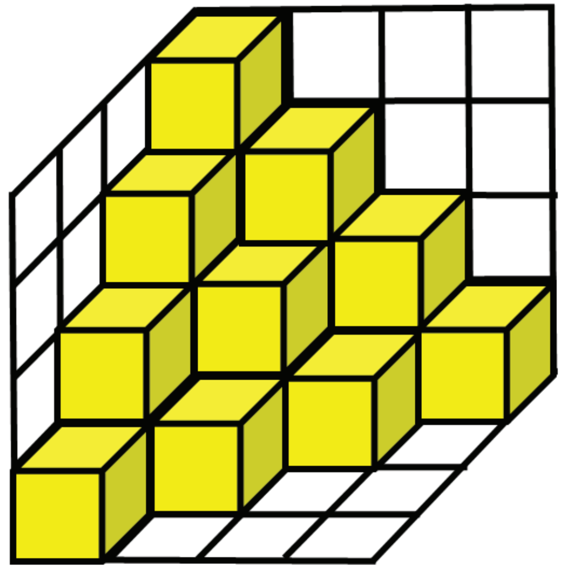


Image credit: High Performance Radiation Transport Simulations: Preparing for TITAN
C. Baker, G. Davidson, T. M. Evans, S. Hamilton, J. Jarrell and W. Joubert, ORNL, USA

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
  - Polyhedral frameworks, only support affine loops, ChiLL and Pluto
- No solution in high-level languages like OpenMP/OpenACC; no software abstractions

Our Contribution: Create Software Abstractions for Wavefront pattern

- Analyzing flow of data and computation in wavefront codes
- Memory and threading challenges
- Wavefront loop transformation algorithm

Abstract Parallelism Model

```
for( iz=izbeg; iz!=izend; iz+=izinc )
for( iy=iybeg; iy!=iyend; iy+=iyinc )
for( ix=ixbeg; ix!=ixend; ix+=ixinc )   { // space

  for( ie=0; ie<dim_ne; ie++ ) { // energy
    for( ia=0; ia<dim_na; ia++ ) {  // angles
      // in-gridcell computation
    }
  }
}
```

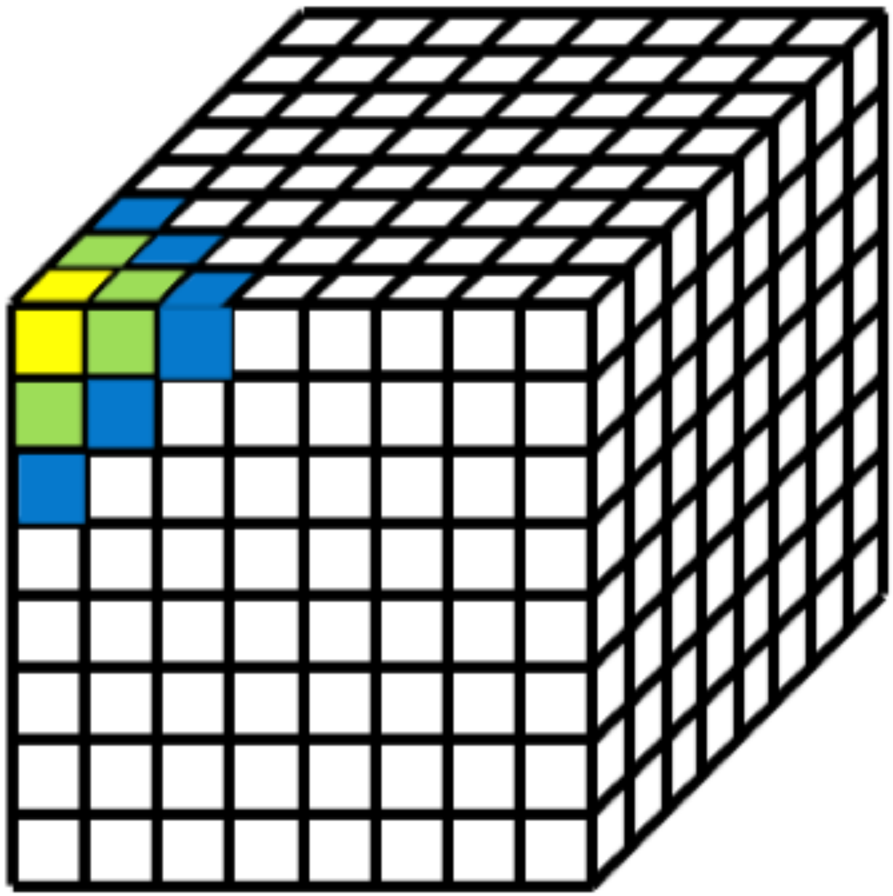
Abstract Parallelism Model

- 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

Data Model

- 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

Parallelizing Sweep Algorithm: KBA



Programming Model Limitations

- No abstraction for wavefront loop transformation
  - Manual loop restructuring
- Limited layers of parallelism
  - 2 isn't enough (worker is broken)
  - Asynchronous execution?

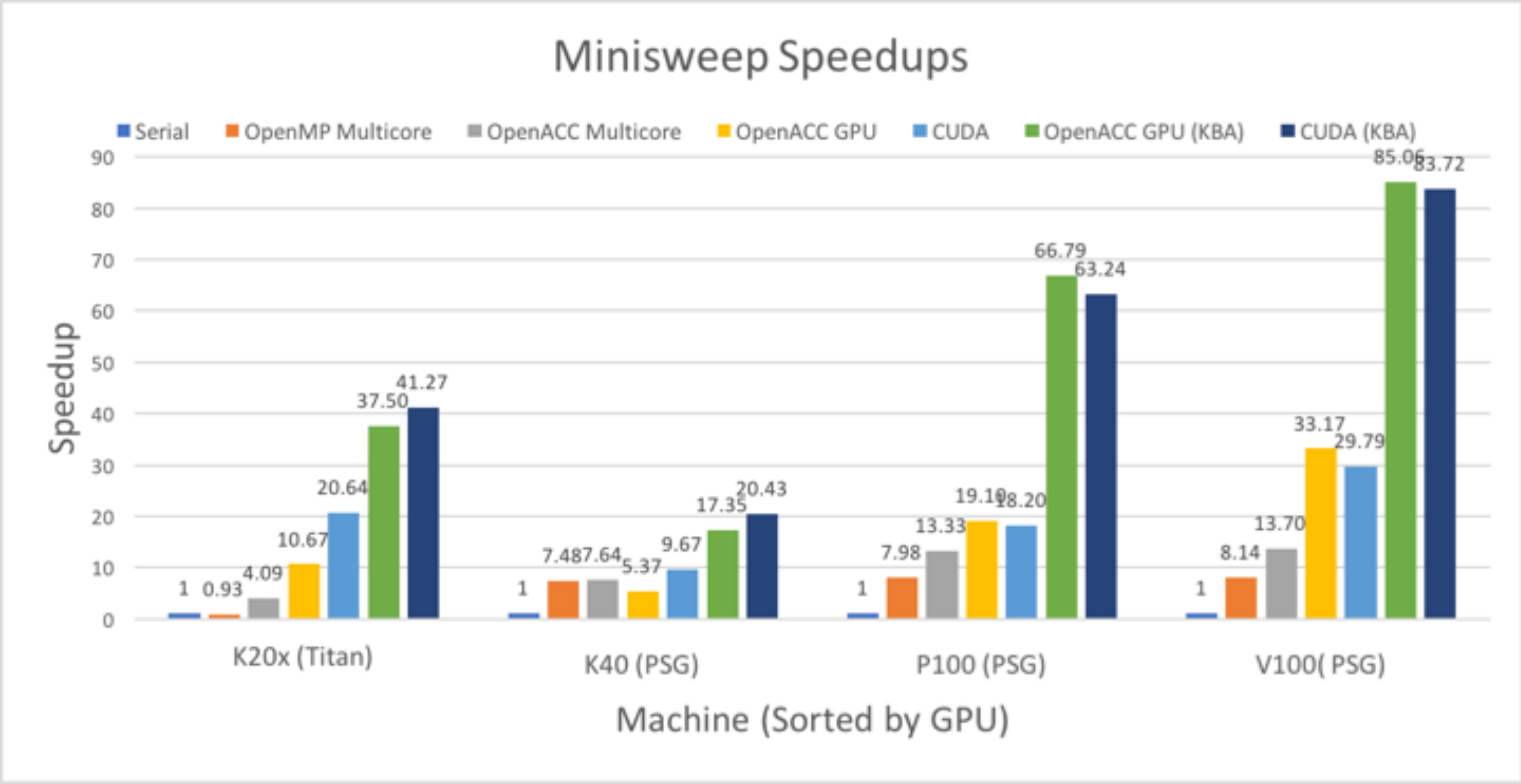
Experimental Setup

- NVIDIA PSG Cluster
  - CPU: Intel Xeon E5-2698 v3 (16-core) and Xeon E5-2690 v2 (10-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
- ORNL SummitDev
  - CPU: IBM Power8 (10-core)
  - GPU: NVIDIA Tesla P100
- PGI OpenACC Compiler 17.10
- OpenMP – GCC 6.2.0
  - Issues running OpenMP minisweep code on Titan but works OK on PSG.

Input Parameters

- Scientifically
  - X/Y/Z dimensions = 64
  - # Energy Groups = 64
  - # Angles = 32
- Goal is to explore larger spatial dimensions

Machine	CPU	GPU
NVIDIA PSG (V100)	Intel Xeon E5-2698 v3 (16 cores)	NVIDIA Tesla V100 (16GB HBM2)
NVIDIA PSG (P100)	Intel Xeon E5-2698 v3 (16 cores)	NVIDIA Tesla P100 (16GB HBM2)
NVIDIA PSG (K40)	Intel Xeon E5-2690 v2 (10 cores)	NVIDIA Tesla K40 (12GB GDDR5)
ORNL Titan	AMD Opteron 6274 (16 cores)	NVIDIA Tesla K20X (6GB GDDR5)



Contributions

- An abstract representation of wavefront algorithms
- A performance portable proof-of-concept of this abstraction using OpenACC
  - Evaluation on multiple state-of-the-art platforms
- A description of the limitations of existing high-level programming models

Next Steps

- Asynchronous execution
- MPI - multi-node/multi-GPU
- Develop a generalization/extension to existing high-level programming models
  - Prototype

Preliminary Results/Ongoing Work

- MPI + OpenACC
  - 1 node x 1 P100 GPU = 66.79x speedup
  - 4 nodes x 4 P100 GPUs/node = 565.81x speedup
  - 4 nodes x 4 V100 GPUs/node = 624.88x speedup
- Distributing the workload lets us examine larger spatial dimensions
  - Future: Use blocking to allow for this on a single GPU

Takeaway(s)

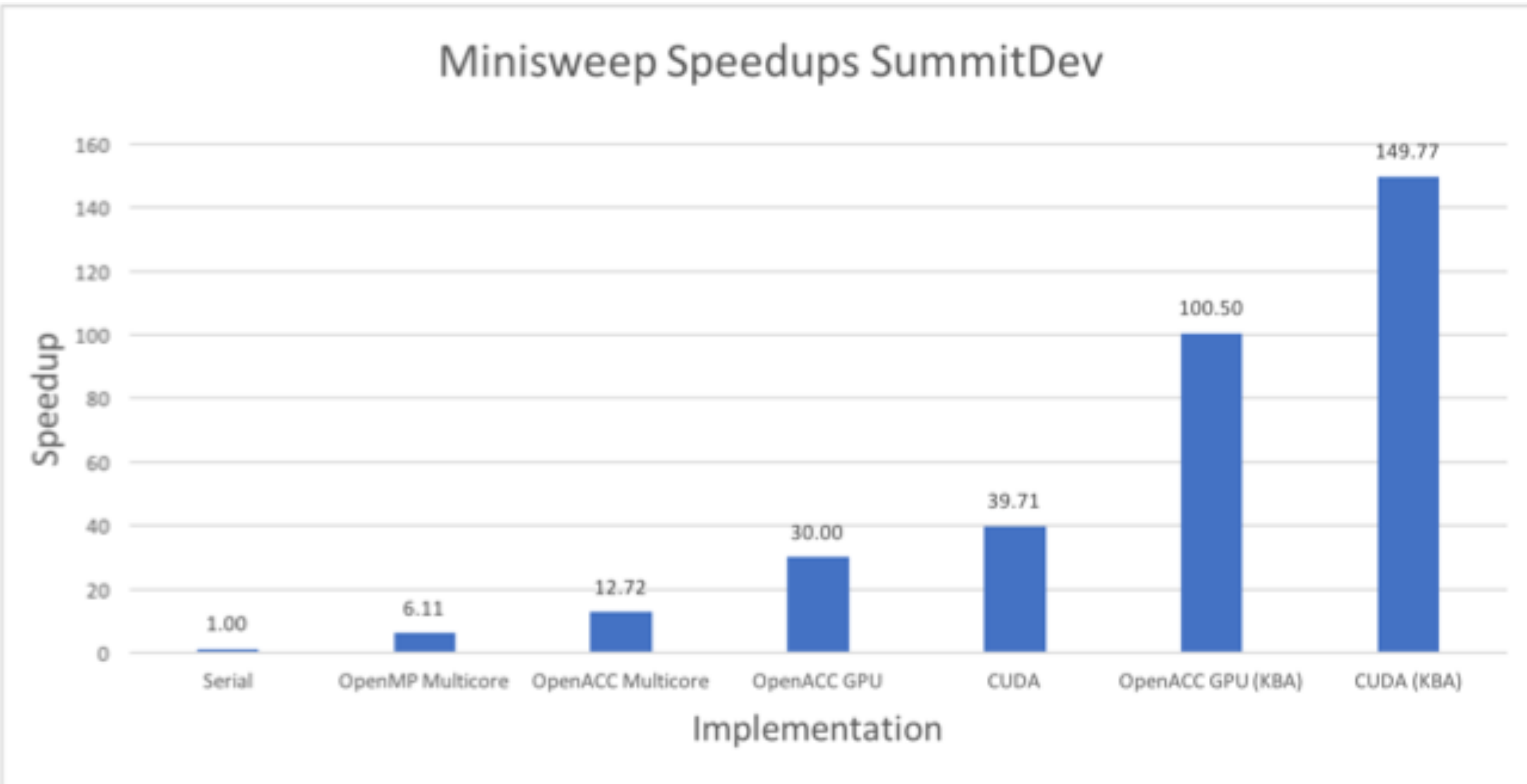
- Using directives is not magical! Compilers are already doing a lot for us! ☺
- Code benefits from incremental improvement – so let's not give up! ☺
- *Profiling and Re-profiling is highly critical*
- Look for any serial code refactoring, if need be
  - Make the code parallel and accelerator-friendly
- Watch out for compiler bugs and *report them*
  - The programmer is not 'always' wrong
- Watch out for *novel language extensions and propose to the committee* - User feedback
  - Did you completely change the loop structure? Did you notice a parallel pattern for which we don't have a high-level directive yet?

Contributions

- An abstract representation of wavefront algorithms
- A performance portable proof-of-concept of this abstraction using directives, OpenACC
  - Evaluation on multiple state-of-the-art platforms
- A description of the limitations of existing high-level programming models
- Contact: rsearles@udel.edu
- Github: https://github.com/rsearles35/minisweep

Additional Material

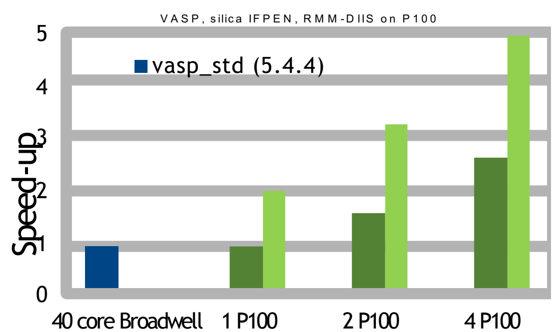
Additional Material



OPENACC GROWING MOMENTUM

Wide Adoption Across Key HPC Codes

ANSYS Fluent, Gaussian, VASP



* OpenACC port covers more VASP routines than CUDA, OpenACC port planned top down, with complete analysis of the call tree, OpenACC port leverages improvements in latest VASP Fortran source base

3 of Top 5 HPC Applications Use OpenACC

GTC
XGC
ACME
FLASH
LSDalton

CAAR Codes Use OpenACC

<u>Key Codes Globally</u>

COSMO, IFS(ESCAPE), NICAM, ICON, MPAS

<u>Gordon Bell Finalist</u>

CAM-SE on Sunway Taihulight

OpenACC Dominates in Climate & Weather

OpenACC

- 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
- Poses a six dimensional problem
- 3D in space, 2D in angular particle direction and 1D in particle energy