SPF: Sparse Polyhedral Framework and Friends

Anna Rift
Boise State University

Ravi Shankar
Boise State University

Tobi G. Popoola
Boise State University
SPF: Sparse Polyhedral Framework and Friends

Abstract
Scientific applications are computationally intensive and require expensive HPC resources. Optimizing scientific applications requires that we balance three competing goals: Performance, Productivity, and Portability. Performance is important because it reduces time to solution and power consumption. However, optimization has the potential to negatively impact scientific productivity due to obfuscating the code. Portable code, code that can be moved to different computers, tends to be slow and difficult to maintain. We explore using the Sparse Polyhedral Framework to create a compiler internal representation that efficiently supports optimization techniques. Automating optimizations will strike a balance among performance, productivity, and portability.
1. Problem Statement
- Scientific applications → Computationally intensive → require expensive HPC resources
- Optimizing scientific applications requires a balance of Performance, Productivity, and Portability.

2. Motivation
Speedup of executor transformed for wavefront parallelism over library serial code.

3. The Polyhedral Model
- Represents the iteration of each statement of a computation in a loop nest as lattice points in a polyhedron.
- Only supports affine data accesses, and does not work for sparse computations.

4. Sparse Data
- Represents the iteration of each statement of a computation in a loop nest as lattice points in a polyhedron.
- For affine data accesses, and does not work for sparse computations.

5. Sparse Polyhedral Framework (SPF)
- Extends the polyhedral model
- Provides a mathematical framework for representing and transforming irregular computations (uninterpreted functions)
- Suitable for non-affine loop bounds present in irregular applications.

6. Optimization Overview
- SPF: Sparse Polyhedral Framework and Friends
- Anna Rift, Ravi Shankar, Tobi Popoola
- Advisor: Dr. Catherine Olschanowsky

7. Intermediate Representation
- Computation IR
- Code Gen
- optimized C code

8. Future Development
- Currently only have an identity transformation, need to write more
- Algorithmically manipulating data layout to meet execution requirements
- Inlining computations that call others
- Synthesize IR to facilitate conversion from one sparse format to another

9. Acknowledgements
Boise State’s Research Computing Department.

10. Collaborators
Boise State’s Research Computing Department.