Abstractions for Specifying Sparse Matrix Data Transformations

Payal Nandy
Mary Hall
University of Utah

Eddie C Davis
Catherine Olschanowsky
Boise State University

Mahdi S Mohammadi, Wei He
Michelle Strout
University of Arizona

MOTIVATION

• The polyhedral model is suitable for affine transformations
• Loop bounds, array access expressions and transformations
• Polyhedral model unsuitable for sparse matrix & unstructured mesh computations (non-affine)
• Array accesses of the form A[i][j]
• Loop bounds of the form index[i] ≤ j < index[i+1]

Key Observation

Compiler generated code for run time inspector & executor

Inspector
• Collects information that is only available at runtime
• Widely used in optimizing irregular computations, where information about data dependences, loop bounds, data structures, and memory access patterns
• Used to guide code transformation, parallelization, and data layout

Executor
• Optimized version of original computation created at compile time.
• Uses optimized data layout and iteration spaces generated by Inspector

CONTRIBUTION

Derive abstractions for Sparse Matrix Data Transformations
• Focus on transformations that modify data representation

Extend Sparse Polyhedral Framework to Support data transformations
• Modify data representation to reflect structure of input matrix
• Expand iteration space to match new data representation
• Goal: automatically compose them

EXAMPLE – CSR to DIA

Transformation Relations

T\text{make-dense} = (\{i,j\} \rightarrow \{i,j\}, 0 \leq k \leq N \wedge k = \text{col}(i))
T\text{keep} = (\{i,k\} \rightarrow \{i,k\}, k = k')
T\text{compact-and-pass} = (\{k,k'\} \rightarrow \{d\}, 0 \leq d < ND \wedge k' = \text{col}(i) \wedge \text{d}(d) = k')
\text{lexec} = T\text{compact-and-pass}(T\text{make-dense}(i))

Generator

D_set = (\{k'| \exists j, k' = \text{col}(j) \wedge \text{index}(i) \leq j < \text{index}(i+1)\})
ND = \text{count}(D\text{-set})
C = \text{order}(D\text{-set})
A\text{-prime} = \text{alloc}(N*ND*\text{sizeof(datatype)})
\text{map}: R \rightarrow A\text{-prime} = \{(i, j) \rightarrow \{l, d\}, 0 \leq d < ND \exists \text{index}(i) \leq j \leq \text{index}(i+1)\}

Executor Code

for(\{i, j\} \in \text{D\text{-set}})
for(\text{d} = \text{lookup}(\text{A\text{-prime[ND]}}, \text{D\text{-set}}))
\text{y}[\text{index}(i) + \text{d}] += A[\text{index}(j)]*x[\text{index}(j) + \text{d}];

CONCLUSION

• Abstractions for data transformations in sparse matrix & unstructured mesh computations
• Vision: Create a framework to compose complex transformation sequences for inspectors and executors
• Minimize inspector passes over input data
• Extend IDG to support fusion of inspectors
• Integrate existing inspector library functions