SpDISTAL: Compiling Distributed Sparse Tensor Computations

\[ a = B \cdot c \]
\[ A = B \cdot C \quad A = B + C \]
\[ A = B + C + D \]
\[ A_{ij} = B_{ijk} \cdot c_k \quad A_{ij} = B_{ij} \cdot C_{ik} \cdot D_{kj} \]

\[ \ldots \quad \ldots \quad \ldots \quad \ldots \]
Predefined set of kernels

\[ a = B \cdot c \]
\[ A = B \cdot C \]
\[ A = B + C \]

Predefined set of formats

CSR, ELL, …

Get close to peak performance (for sparse kernels)!
Higher order data?

Kernels outside of predefined set?

\[
A = B + C + D
\]
\[
A_{ij} = B_{ijk} \cdot c_k \\
A_{ij} = B_{ij} \cdot C_{ik} \cdot D_{kj}
\]

Flexible data formats?
Cyclops Tensor Framework
parallel arithmetic on multidimensional arrays

All of tensor algebra

C["..."] += A["..."] * B["..."]

Leaves performance on the table

1-2 orders of magnitude slower than hand-written
Programmers today face a trade-off!

Libraries of Kernels
- PETSc
- TAO
- TRILINOS

Performance

Generality
(All of tensor algebra)

Compilation
This work: SpDISTAL

Factorization
Cyclops Tensor Framework (CTF)
What to compute
(*Tensor Index Notation*)

How data is compressed
(*Format Language*)

How computation is distributed
(*Scheduling Language*)

How data is distributed
(*Tensor Distribution Notation*)

\[ y(i) = A(i, j) \times x(j) \]

x, y distributed in chunks
A distributed row-wise

SpDISTAL
Enables writing distributed sparse programs in high level languages

Separates concerns of correctness and performance

Accelerates development of distributed sparse tensor programs
What to compute
(Tensor Index Notation)

How data is compressed
(Format Language)

How computation is distributed
(Scheduling Language)

How data is distributed
(Tensor Distribution Notation)

SpDISTAL

DISTAL: The Distributed Tensor Algebra Compiler

Rohan Yadav
Stanford University
Stanford, CA, USA
rohany@cs.stanford.edu

Alex Aiken
Stanford University
Stanford, CA, USA
aiken@cs.stanford.edu

Fredrik Kjolstad
Stanford University
Stanford, CA, USA
kjolstad@stanford.edu
\[ y(i) = A(i, j) \times x(j) \]

\( x = y = \{ \text{Dense} \} \)
\( A = \{ \text{Dense, Sparse} \} \)

\( \text{distribute}(i) \)

\( x, y \) distributed in chunks
\( A \) distributed row-wise

**SpDISTAL**

Partitioning and communication analysis

Compute kernel generation

Program using Legion Runtime API

**Legion: Expressing Locality and Independence with Logical Regions**

<table>
<thead>
<tr>
<th>Michael Bauer</th>
<th>Stanford University</th>
<th><a href="mailto:mebauer@cs.stanford.edu">mebauer@cs.stanford.edu</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sean Treichler</td>
<td>Stanford University</td>
<td><a href="mailto:sj@cs.stanford.edu">sj@cs.stanford.edu</a></td>
</tr>
<tr>
<td>Elliott Slaughter</td>
<td>Stanford University</td>
<td><a href="mailto:slaughter@cs.stanford.edu">slaughter@cs.stanford.edu</a></td>
</tr>
<tr>
<td>Alex Aiken</td>
<td>Stanford University</td>
<td><a href="mailto:aiken@cs.stanford.edu">aiken@cs.stanford.edu</a></td>
</tr>
</tbody>
</table>
Generation of data partitioning code is the key challenge

Decompose partitioning into a per-dimension problem

Use hybrid static and dynamic analyses

**Static:** generate code that describes the structure of data partitions

**Dynamic:** at runtime, use dynamic analysis to compute precise partitions
Intuition: Coordinate Trees

{Dense, Sparse}
Partitioning Coordinate Trees

1) Create an initial partition of a level
2) Derive a full partition of the tree
size 4
pos 0-2 3-4 5-5 6-7
valr 0 1 3 1 3 0 0 3
vals a b c d e f g h
Partitioning Operators

tiling(R, tile_size)

image(R1, R2)

preimage(R1, R2)

Dependent Partitioning

Sean Treichler
Stanford University, USA
sjt@cs.stanford.edu

Michael Bauer
NVIDIA Research, USA
mbauer@nvidia.com

Rahul Sharma
Stanford University, USA
sharma@cs.stanford.edu

Elliott Slaughter
Stanford University, USA
slaughter@cs.stanford.edu

Alex Aiken
Stanford University, USA
aiken@cs.stanford.edu
Partitioning Sparse Tensors

How can we generate these partitioning calls?
Data Format Describes Partitioning

Formats implement a compile-time interface describing partitioning capabilities

```typescript
type Format {
  ir::Stmt initial_partition(…)
  ir::Stmt partition_from_parent(…)
  ir::Stmt partition_from_child(…)
  … <and more> …
}
```
void RowSplitSpMV(y, A, x) {
    
    A_pos_part = tiling(A.pos, …)
    A_crd_part = image(A_pos_part, A.crd)
    A_val_part = image(A_pos_part, A.val)
    
    task launch (A_pos_part, A_crd_part, A_val_part) {
        < … Optimized SpMV Kernel … >
    } 
}
Describing data partitioning is the key problem

Tackle data partitioning through a per-dimension approach

Utilize hybrid static + dynamic analyses
SpDISTAL achieves performance competitive with hand-tuned systems

SpDISTAL outperforms general factorization-based systems

Run on Lassen (Power9 CPU, 4 V100’s, Infiniband), tensors from SuiteSparse, FROSTT and Freebase

<table>
<thead>
<tr>
<th>Operation</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpMV</td>
<td>$a(i) = B(i, j) * c(j)$</td>
</tr>
<tr>
<td>SpMM</td>
<td>$A(i, j) = B(i, k) * C(k, j)$</td>
</tr>
<tr>
<td>SpAdd3</td>
<td>$A(i, j) = B(i, j) + C(i, j) + D(i, j)$</td>
</tr>
<tr>
<td>SpTTV</td>
<td>$A(i, j) = B(i, j, k) * c(k)$</td>
</tr>
<tr>
<td>SDDMM</td>
<td>$A(i, j) = B(i, j) * C(i, k) * D(k, j)$</td>
</tr>
<tr>
<td>SpMTTKRP</td>
<td>$A(i, l) = B(i, j, k) * C(j, l) * D(k, l)$</td>
</tr>
</tbody>
</table>
SpDISTAL is 1.1x faster

Median 1.2x speedup

SpDISTAL wins 28/38

Median 0.9x slowdown (when losing)

PETSc is 1.2x faster

Median 0.3x slowdown

Only Trilinos succeeded

Trilinos

PETSc

Protocols

OOM

SpDISTAL

SpDISTAL (Batched)

Trilinos

PETSc

OOM
```python
import cunumeric as np
global legate
import legate.sparse as sparse

d = sparse.diags(…)
y = np.ones(A.shape[1])
x = sparse.linalg.cg(A, y)
assert np.allclose(A @ x, y)
```
Legate Sparse: Distributed Drop-in scipy.sparse

Common matrix formats: CSR, CSC, COO, DIA

Arithmetic and linear algebra operations

Parallel format conversion and sparse matrix construction utilities

Iterative solvers, RK integration and more!

https://github.com/nv-legate/legate.sparse
Legate Sparse Implementation

For “Tensor Algebra”-like computations, utilize DISTAL MatVec, Matrix-Matrix Multiplication, Addition, etc.

Easily extend beyond scipy.sparse, to higher-order operations like SDDMM

Leverage cuNumeric + hand-written code for auxiliary operations.
SpDISTAL’s compilation-based approach can achieve high performance and generality for sparse tensor computations on distributed CPU and GPU machines.

Contact: rohany@cs.stanford.edu

rohany.github.io