Accelerating Sparse Tensor Contractions

Saurabh Raje

Advisor: P. (Saday) Sadayappan

Kahlert School of Computing, University of Utah

Existing Gap

- Lack of fast kernels for sparse tensor times sparse tensor (SpTC) contraction.
- Need better compiler optimizations for chains of such contractions - sparse tensor networks.

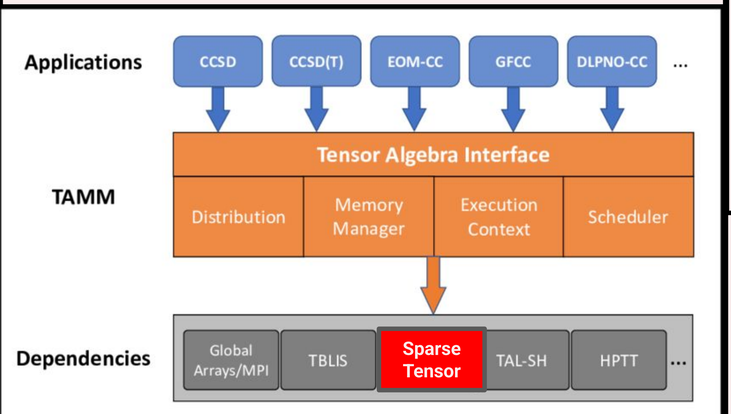
Research contributions

This thesis closes the above gap by developing:

1. **CoNST - Compiler (IR) optimizations for chains of contractions.**
2. **FaSTCC - Library of fast SpTC kernels for CPUs.**
3. **SparseTAMM: A sparse backend to TAMM for computational chemists.**

SparseTAMM

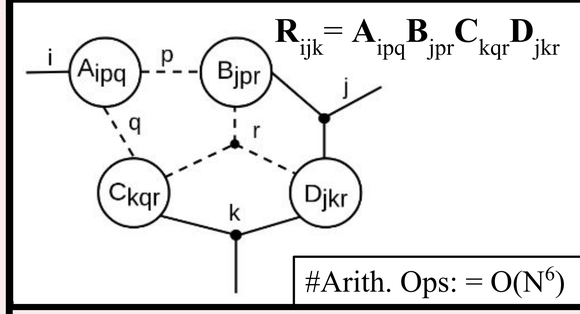
- **Computational chemistry methods have imposed sparsity.**
- **Tensor Algebra for Many body Methods is a system that runs dense tensor contractions.**
- **We add a CPU Sparse backend to this system for chemists to leverage sparsity in tensor contractions.**



- **Energy tensors have pattern sparsity - small dense regions within a large sparse space.**
- **The SparseTensor backend runs contractions over sparse tensors with dense data-types.**
- **Supports any extended einsum expression (with batch indices.)**

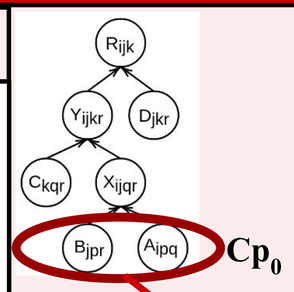
CoNST: Code Generator for Sparse Tensor Networks

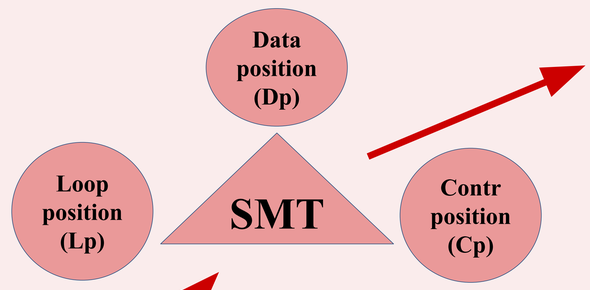
Sparse Tensor Networks



Key contributions

- **Reorder & fuse sparse loops to make intermediate tensors smaller.**
- **Use SMT solver to co-optimize loop and mode order.**
- **Lower to TACO IR with workspaces to accelerate each contraction.**

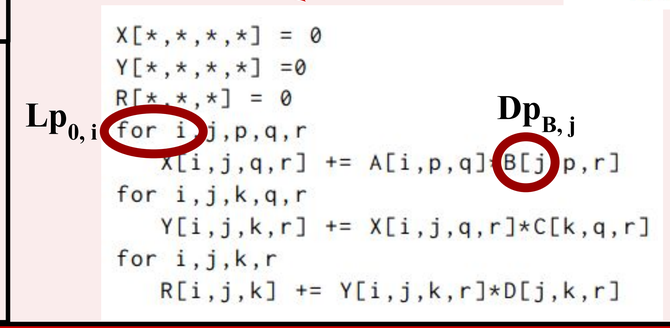


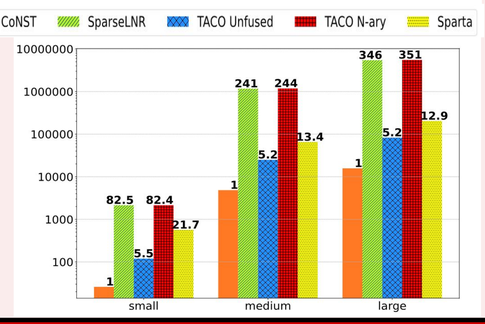



```
for r,j
  for p,q,i
     X[q,i] += A[p,q,i]*B[r,j,p]
  for q,k,i
     Y[k,i] += X[q,i]*C[r,q,k]
  for k,i
     R[j,k,i] += Y[k,i]*D[r,j,k]
```

TACO IR

```
forall(r, forall(j,
where(forall(k, forall(i, R(j, k, i) = Y(k, i) * D(r, j, k))),
      where(forall(q, forall(k, forall(i, Y(k, i) = X(q, i) * C(r, q, k)))),
       forall(p, forall(q, forall(i, X(q, i) = A(p, q, i) * B(r, j, p))))))))
```

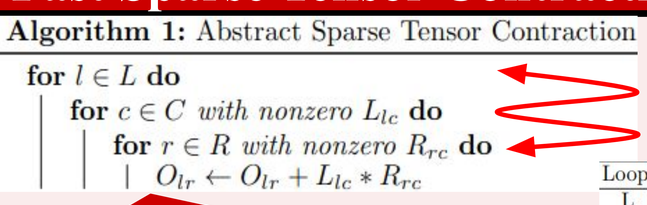




FaSTCC: Fast Sparse Tensor Contractions on CPUs

Key contributions

- **CPU kernel for sparse tensor times sparse tensor.**
- **Tile the contraction to bound workspace size.**
- **Use SOA hashmaps to co-iterate faster.**

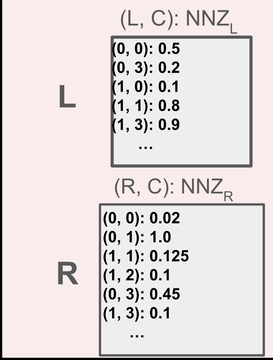


Reordering loops changes the complexity

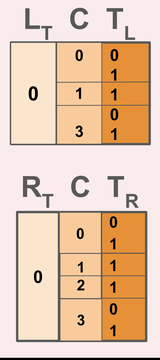
Loop Order	Queries	Volume of Data Moved	Accumulator Size
L, R, C	$O(L \times R)$	$O(L \times nnz_R + R \times nnz_L)$	1
L, C, R	$L + nnz_L$	$O(nnz_L + \frac{nnz_R \times nnz_L}{C})$	R
C, L, R	$O(2 \times C)$	$nnz_L + nnz_R$	$L \times R$

Tile the output to bound accumulator size!

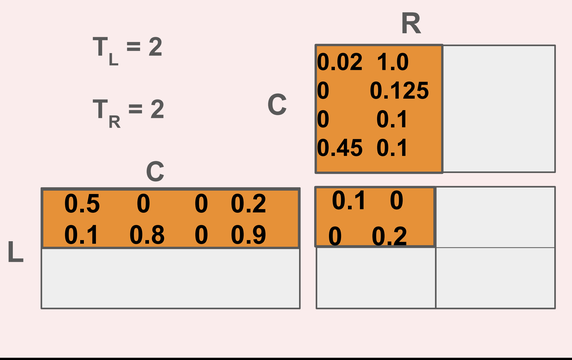
Index (hash) inputs as set of tiles



Iterate over left and right tiles



Thread local Tiled Accumulator for output



Append to COO result

Contraction outer fastest

Needs large accumulator!