Efficient Search-Space Pruning for Integrated Fusion and Tiling Transformations

Xiaoyang Gao, Sriram Krishnamoorthy, Swarup Kumar Sahoo, Chi-Chung Lam, P. Sadayappan
Ohio State University

Gerald Baumgartner, J. Ramanujam,
Louisiana State University

Introduction

- Integrated framework to determine a variety of loop transformations:
  - Loop fusion
  - Loop tiling
  - Loop permutation

- Concrete performance models

- Reduction in the space of possible solutions

Context

- **Tensor Contraction Engine (TCE):** A domain-specific compiler used in Quantum Chemistry.
  - Transform high-level math. specification to efficient parallel programs optimized for target machines.

- **Input:**

- Sequence of tensor contraction expressions

- **Output:**

- Parallel Fortran code

Four-index Transform

$$B(a,b,c,d) = \sum_{p,q,r,s} C1(d,s) * C2(c,r) * C3(b,q) * C4(a,p) * A(p,q,r,s)$$

Operation-minimal form

$$T1(a,q,r,s) = \sum_{p} C4(a,p) \times A(p,q,r,s)$$

$$T2(a,b,r,s) = \sum_{q} C3(b,q) \times T1(a,q,r,s)$$

$$T3(a,b,c,s) = \sum_{r} C2(c,r) \times T2(a,b,r,s)$$

$$B(a,b,c,d) = \sum_{s} C1(d,s) \times T3(a,b,c,s)$$

Producer-consumer relationship

Observations

- Sequence of fully permutable loop nests
- Often, arrays are too large to fit into physical memory
- Array access expressions are loop indices
- In each contraction, indices form three disjoint groups, each group appearing in exactly two array references
  - C[i,j] += A[i,k] * B[k,j]
  - T[i,j] += A[k,l] * B[i,j,k,l]
- A producer loop nest cannot be fused with consumer if summation index is the outermost loop in the producer.

Problem Statement

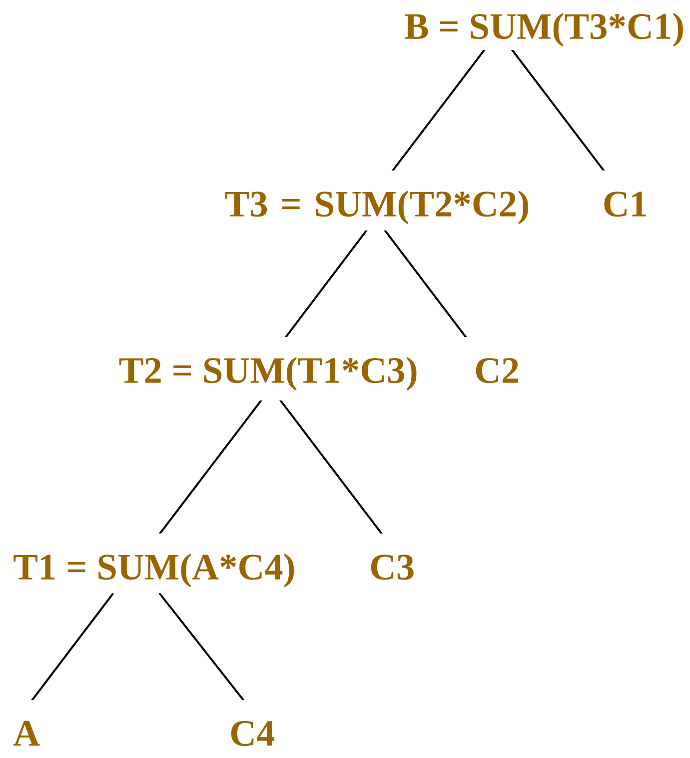
- Objective: Given a tensor expression and machine parameters, determine the appropriate loop transformations, and the position and ordering of I/O placements to minimize disk I/O cost.

- Problem Addressed:
  - Several loop transformations are applied.
  - Their effects on I/O cost are interrelated.
  - Space of possible solutions too large to exhaustively search

- Approach: Pruning of the search space to achieve better solution per effort expended.
- In this paper, we focus on the integration of loop fusion and tiling.

Operation Tree

- Operation Tree: A binary tree represents a sequence of tensor contractions.
  - Leaf: Input arrays
  - Root: Output array
  - Interior node: Intermediate or output arrays, produced by the tensor contraction of their immediate children
  - Edge: *Producer-consumer* relationship between tensor contractions



Problem Statement

- Input : Operation Tree
- Output: Candidate loop structures
- Objective: Minimize number of loop structures to be considered while maximizing search space explored.

Fusion Enumeration Space

- A natural approach
  - All combinations of common loops in related loop nests (producers and consumers in a contraction)
  - Very large solution space.
- Key observation
  - Given any fused structure
    - A canonical fusion structure can be generated
    - All common loops in the loop nests are fused
    - All loops are tiled and tile sizes set appropriately

Two-index Transform

```
for i
   for j,n
      T[n] += A[i,j]*C2[n,j]
  for m,n
       B[m,n] += T[n]*C1[m,i]

for n
   for j,i
      T[i] += A[i,j]*C2[n,j]
  for m,i
       B[m,n] += T[i]*C1[m,i]

for i,n
   for j
      T += A[i,j]*C2[n,j]
  for m
       B[m,n] += T*C1[m,i]
```

T[i,n] = A[i,j] * C2[n,j]

B[m,n] = T[i,n] * C1[m,i]

```
for it1, nt1
   for j, it2, nt2
      T[it2, nt2] += A[it1+it2, j] * C2[nt1+nt2, j]
   for m, it2, nt2
       B[m, nt1+nt2] += T[it2,nt2] * C1[m, it1+it2]
```

Fuse all common loops

Two-index Transform (Contd.)

```
for i
    for j,n
        T[n] += A[i,j]*C2[n,j]
   for m,n
         B[m,n] += T[n]*C1[m,i]
```

```
for it1, nt1=1
    for j, it2=1, nt2
        T[it2, nt2] += A[it1+it2, j] * C2[nt1+nt2, j]
    for m, it2=1, nt2
         B[m, nt1+nt2] += T[it2,nt2] * C1[m, it1+it2]
```

Fusion + tiling to reduce number of candidate loop structures

Cut-point and Fused Sub-tree

- To fuse or not-to-fuse

- Cut-point: For a fusion structure, an intermediate node not fused with its consumer, is a *cut-point* in the operation tree.

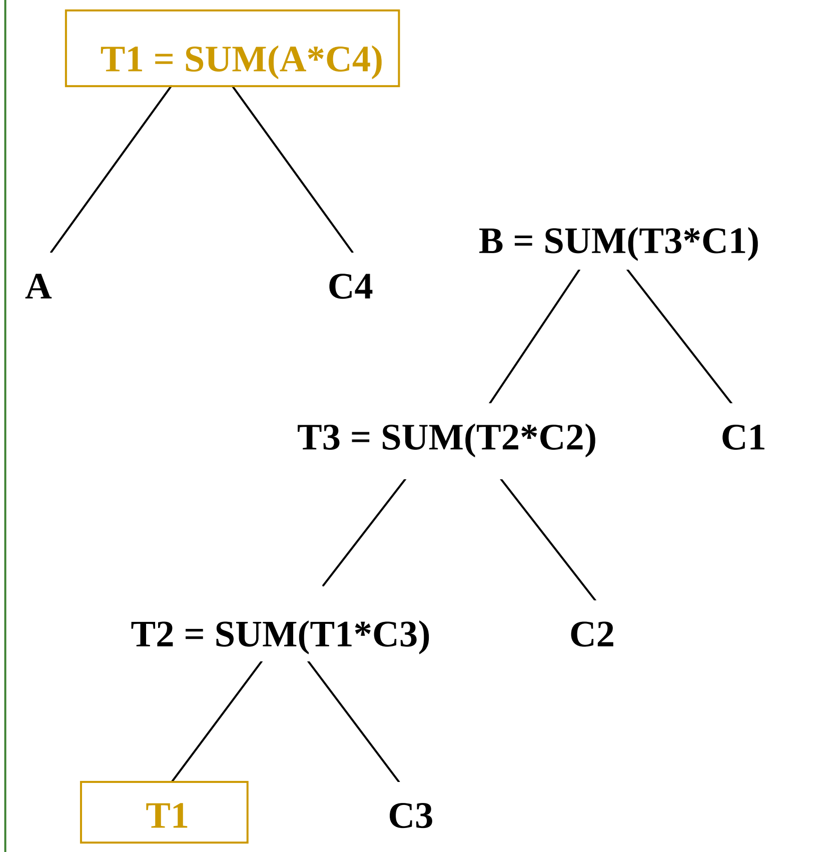
- Fused Sub-tree: Cut-points divide an operation tree into several sub-trees. A sub-tree without any interior cut-points is a *fused sub-tree*.

Fused Sub-tree and Cut-point (4index)

Loop Structure:

```
for a,r,q,s,p
  T1(a,q,r,s)+=A(p,q,r,s)*C4(a,p)

for a,b
  for r,s
    for q
      T2(r,s)+=T1(a,q,r,s)*C3(b,q)
    for c
      T3(c,s)+=T2(r,s)*C2(c,r)
 for c,d,s
    B(a,b,c,d)+=T3(c,s)*C1(d,s)
```



Integrated Framework

Input: Operation Tree

Procedure:

- Operation Tree Partitioning
- Loop Structures Enumeration
- Intra-Tile Loop Placements
- Disk I/O Placements and Orderings
- Tile Size Selection
- Code Generation

Output: Fortran Code

Operation Tree Partitioning

- Partition the operation tree using cut-points
- Each intermediate tree node is potentially a cut-point
- Operation tree with *M* intermediate nodes – 2^M fusion structures

Fused Sub-tree Enumeration

- Three choices for each contraction
  - Fuse all loops common to any two of the three nodes involved in the contraction
    - The two producer nests and the consumer nest
- Fusing the loops of the producer loop-nests places the summation indices as the outermost
  - Fusion structure cannot be extended – a cut-point

All fusion sub-structures to be enumerated are chains

Fused Sub-tree Enumeration

- Dynamic programming solution to construct fusion structures hierarchically
  - At any interior node of operation tree,
    - Extend fusion structures of the producer nests to the consumer or
    - Fuse the loops of the producer and terminate the fusion structure.

Loop Structure Enumeration

1. Fusion sub-trees form a chain of contractions.
2. All possible enumerations of loop structures - *parenthesization* problem
3. For each parenthesization, a *maximally fused loop structure* is created by a recursive construction procedure.
   - *Maximally fused loop:* Each loop nest in which two subnest have as many common loops as possible.

Maximally fused loop structure

1. 4index: $B(a,b,c,d) = \sum_{p,q,r,s} C1(d,s) * C2(c,r) * C3(b,q) * C4(a,p) * A(p,q,r,s)$
2. Contraction sequence:

$$T1(a,q,r,s) = \sum_{p} C4(a,p) * A(p,q,r,s)$$

$$T2(a,b,r,s) = \sum_{q} C3(b,q) * T1(a,q,r,s)$$

$$T3(a,b,c,s) = \sum_{r} C2(c,r) * T2(a,b,r,s)$$

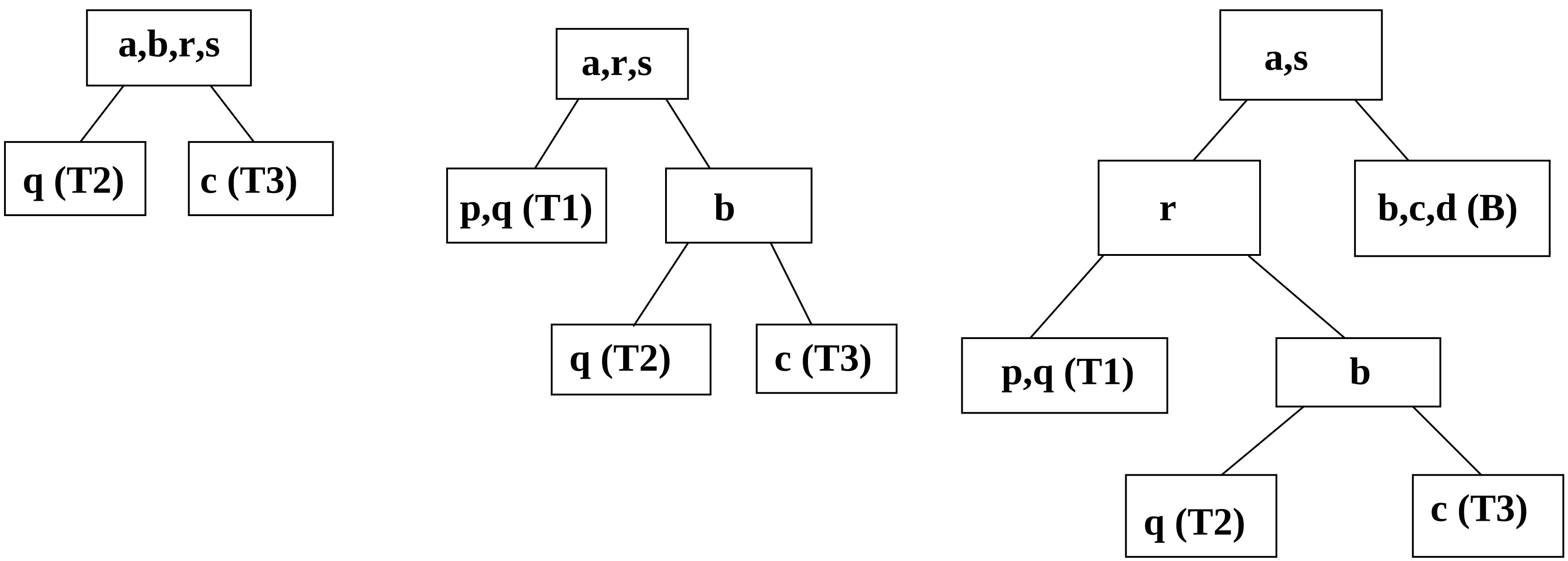
$$B(a,b,c,d) = \sum_{s} C1(d,s) * T3(a,b,c,s)$$

3. Contraction chain: T1 T2 T3 B
4. Parenthesizations: (T1(T2(T3B))), ((T1(T2T3))B), (T1((T2T3)B)), (((T1T2)T3)B) , ((T1T2)(T3B)), (T1(T2(T3B)))

Maximally fused loop structure (Contd.)

5. Maximally fused loop structure for ((T1(T2T3))B):

(T2T3) (T1(T2T3)) ((T1(T2T3))B)



Experimental Evaluation

- Determined the reduction in the number of possible loop structures before and after pruning.
- Evaluated on representative expressions from three quantum chemistry codes:
  - Four-index transform (4index)
  - CCSD computation (CCSD)
  - CCSDT computation (CCSDT)

Experimental Evaluation

Expressions	Total loop structures	Loop structures after pruning	Reduction
4index	241	5	98%
CCSD	69	2	97%
CCSDT	182	5	98%

Conclusions

- Partitioned an operation tree into fused sub-trees.
- Determined candidate loop structures as parenthesizations of candidate fusion chains.
- Search space of possible loop structures is drastically reduced.

Thank You!