

•Analytical Cache Modeling and Tile-size Optimization for Tensor Contractions

Rui Li¹, Aravind Sukumaran-Rajam², Richard Veras³, Tze-Meng Low⁴, Fabrice Rastello⁵, Atanas Rountev⁶, P. Sadayappan¹

Utah¹, WSU², LSU³, CMU⁴, INRIA⁵, OSU⁶

Overview

Goal

- High performance tensor contraction on CPU

Challenge

- Exponential space of valid code configurations
- How to select the best performing code configuration?

Solution

- Data movement is the critical factor affects performance
- Develop analytical models to estimate the data movement based on the code configuration
- Estimate the time of each code configuration and select the best performing version

Contributions

- Modeling data movement of a multi-level memory hierarchy system as a non-linear optimization problem
- Estimating the execution time of tensor contractions on a multi-level memory system

Data Movement Modeling for Tiled Loop Code

6D Tiled Loop Code

- 6! Permutations brought by 6 tiling loops

- Tile sizes are selected such that the data foot print fits in cache

- Permutation of point loops has no effects to data movement

```
for (ti=0:Ni:Ti)
  for (tj=0:Nj:Tj)
    for (tk=0:Nk:Tk)
      for (tl=0:Nl:Tl)
        for (tm=0:Nm:Tm)
          for (tn=0:Nn:Tn)
            for (i=ti:ti+Ti-1)
              for (j=tj:tj+Tj-1)
                for (k=tk:tk+Tk-1)
                  for (l=tl:tl+Tl-1)
                    for (m=tm:tm+Tm-1)
                      for (n=tn:tn+Tn-1)
                        C[i][j][k][l] +=
                          A[i][m][k][n] *
                          B[j][n][l][m];
```

List.1 Tiled version of tensor contraction (single level cache), Ni, Nj, ..., Nn are problem ranges, Ti, Tj, ..., Tn are tiles sizes.

Automatic Data Movement Calculation

- Bottom-up algorithm for each permutation
- Prune loop permutations with same data movement expression

```
for (ti=0; ti<Ni; ti+=Ti)
  for (tj=0; tj<Nj; tj+=Tj)
    for (tk=0; tk<Nk; tk+=Tk)
      for (tl=0; tl<Nl; tl+=Tl)
        for (tm=0; tm<Nm; tm+=Tm)
          for (tn=0; tn<Nn; tn+=Tn)
            tiles of Cijkl, Aimkn, Bjnlm
```

Problem Formalization for Single Level Cache

- Conditional Optimization problem based on data movement expression and cache capacity constraint

- argmin_{all T} $N_i N_j N_k N_l + N_i N_m N_k N_n \left(\frac{N_l}{T_l}\right) \left(\frac{N_j}{T_j}\right) + N_j N_n N_l N_m \left(\frac{N_k}{T_k}\right) \left(\frac{N_i}{T_i}\right)$
- under constraint $T_i T_j T_k T_l + T_i T_m T_k T_n + T_j T_n T_l T_m \leq C$

Edge Cases Handling: Tile Size Equal to Range

- If some problem range N_x can fit in cache, setting $T_x = N_x$ changes the cost
- 2^d cases to be traversed for a pruned permutation Must be performed within and across trees
- Fig.1 shows all cases for a fixed permutation of Matmul
- Each leaf contains a data movement expression
- Merge leaves of all trees with same cost expression

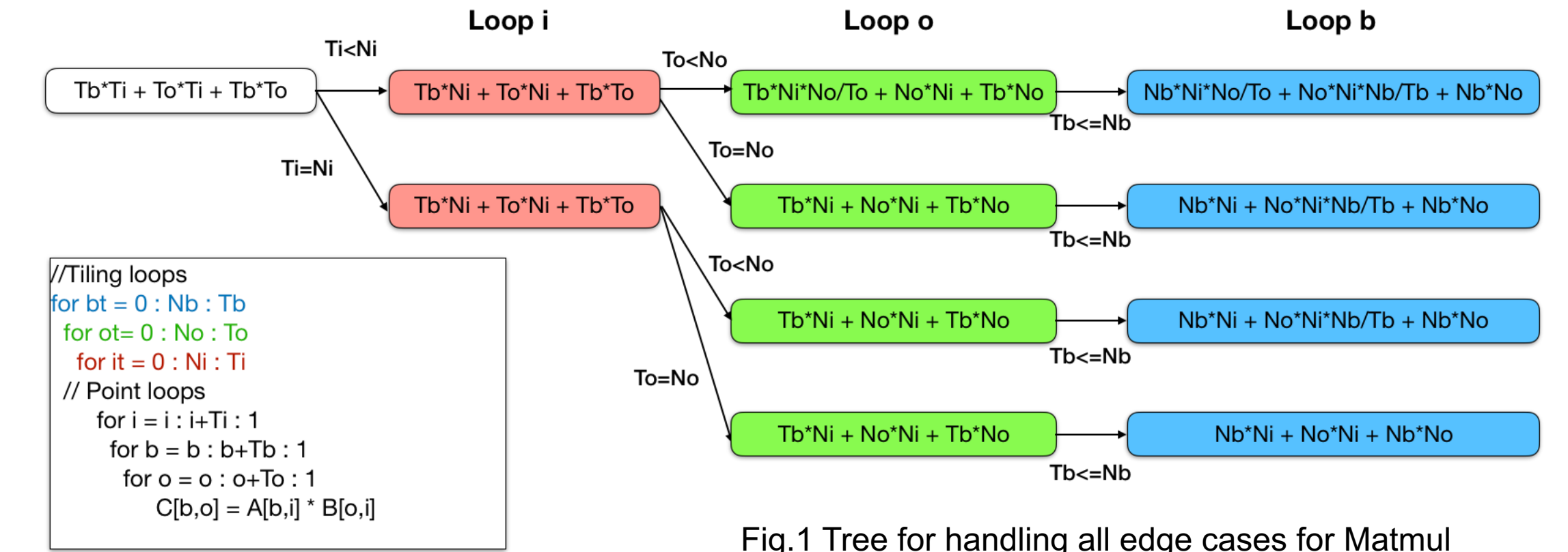
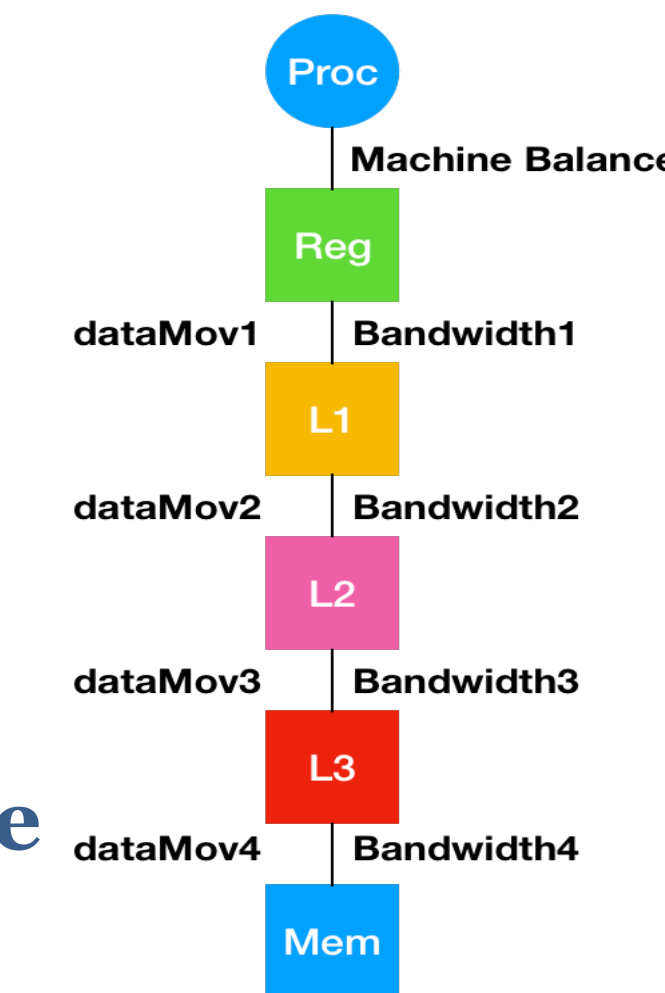


Fig.1 Tree for handling all edge cases for Matmul

Time Cost Model on Multi-level Cache



- Optimization Problem: $\arg \min_T \left(\max_{i=1,2,3,4} \frac{dataMov_i}{bandwidth_i} \right)$
- P^L different max-min problems to solve for L-level memory hierarchy system, ($P = \#Pruned \text{ permutations}$)
- Solver: Couenne (from COIN-OR)
- Utilize BLIS micro-kernel to guarantee high-throughput on FMA unit
- Packing is required to reduce conflict misses

Experimental Evaluation

- Geometric Mean speedup is 1.34x versus TBLIS, 1.27x versus TCL-MKL, and 1.34x versus TCL-BLIS on i7-6700K
- Performance model can also be used for predicting execution time (less than 10% for most cases)

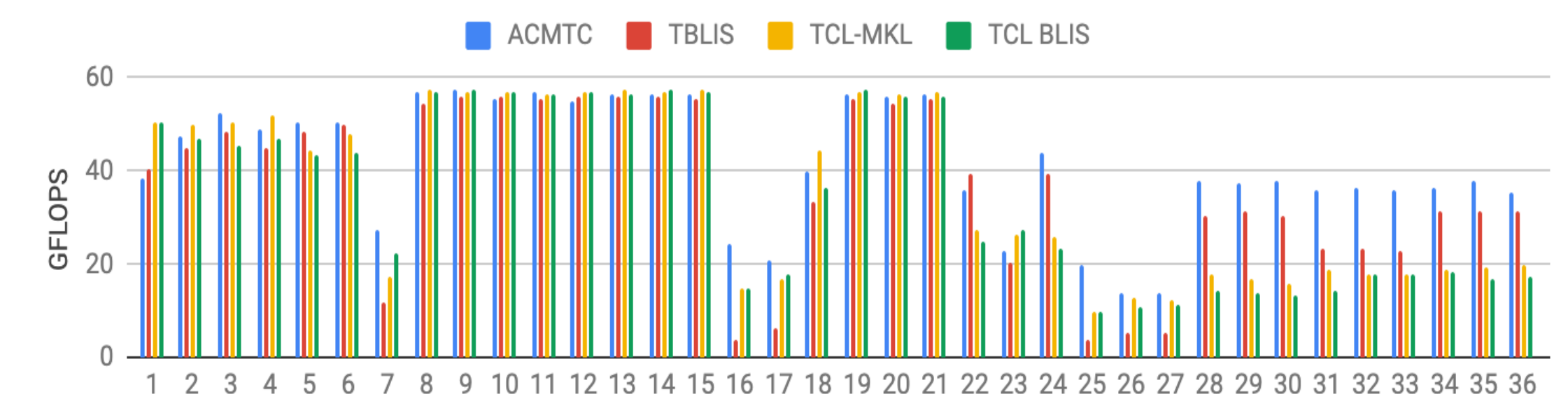


Fig.2 Evaluation on TCG benchmark on i7-6700K, single thread, comparing to TBLIS, TCL-BLIS and TCL-MKL