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

Rui Li[1], Aravind Sukumaran-Rajam[2], Richard Veras[3], Tze-Meng Low[4], Fabrice Rastello[5], Atanas Rountev[6], P. Sadayappan[1]

Utah[1], WSU[2], LSU[3], CMU[4], INRIA[5], OSU[6]

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

.Footprints in point loops fit in cache

.Permutation of point loops has no effects to data movement

```
for(ti=0:Ni:Ti)
 for(tj=0:Ni: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

$$N_iN_jN_kN_l + N_iN_mN_kN_n(\frac{N_l}{T_l})(\frac{N_j}{T_j}) + N_jN_nN_lN_m\left(\frac{N_k}{T_k}\right)(\frac{N_i}{T_i})$$

```
for (ti=0; ti<Ni; ti+=Ti)
```
$$T_iN_jN_kN_l + T_iN_mN_kN_n(\frac{N_l}{T_l})(\frac{N_j}{T_j}) + N_jN_nN_lN_m(\frac{N_k}{T_k})$$
```
 for (tj=0; tj<Nj; tj+=Tj)
```
$$T_iT_jN_kN_l + T_iN_mN_kN_n(\frac{N_l}{T_l}) + T_jN_nN_lN_m(\frac{N_k}{T_k})$$
```
  for (tk=0; tk<Nk; tk+=Tk)
```
$$T_iT_jT_kN_l + T_iN_mT_kN_n(\frac{N_l}{T_l}) + T_jN_nN_lN_m$$
```
   for (tl=0; tl<Nl; tl+=Tl)
```
$$T_iT_jT_kT_l + T_iN_mT_kN_n + T_jN_nT_lN_m$$
```
    for (tm=0; tm<Nm; tm+=Tm)
```
$$T_iT_jT_kT_l + T_iT_mT_kN_n + T_jN_nT_lT_m$$
```
     for (tn=0; tn<Nn; tn+=Tn)
```
$$T_iT_jT_kT_l + T_iT_mT_kT_n + T_jT_nT_lT_m \le C$$
```
      tiles of C_ijkl, A_imkn, B_jnlm
```

Problem Formalization for Single Level Cache

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

•$\underset{all\ T}{\text{argmin}}\ N_iN_jN_kN_l + N_iN_mN_kN_n(\frac{N_l}{T_l})(\frac{N_j}{T_j}) + N_jN_nN_lN_m\left(\frac{N_k}{T_k}\right)(\frac{N_i}{T_i})$

• under constraint $T_iT_jT_kT_l + T_iT_mT_kT_n + T_jT_nT_lT_m \le C$

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

•Merging leaves of all trees with same cost expression

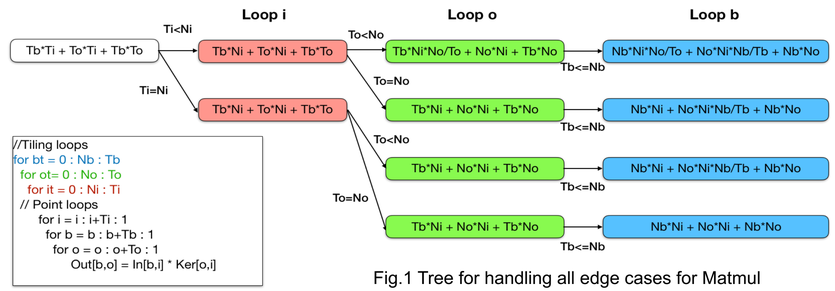
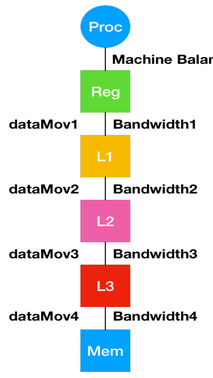


Fig.1 Tree for handling all edge cases for Matmul

Time Cost Model on Multi-level Cache



•Optimization Problem: $\arg\min_{T}(\max_{i=1,2,3,4}\frac{dataMov_i}{bandwidth_i})$

•P^L different max-min problems to solve for L-level memory hierarchy system, $(P = \#Pruned\ permutations)$

•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

•Evaluation on TTCG benchmark on i7-6700K, single thread, comparing to TBLIS, TCL-BLIS and TCL-MKL

•Competitive when all tensors are huge

•Outperform when some inputs tensor are large or output tensor is large