Understanding Memory Effects in the Automated Generation of Optimized Matrix Algebra Kernels

Liz Jessup

Joint work with
Ian Karlin, Erik Silkensen,
Geoffrey Belter and Jeremy Siek

University of Colorado at Boulder
The project

• Build to Order (BTO) compiler to automate loop fusion for matrix algebra kernels
The motivation

• Matrix algebra is essential in many important applications
• Flop speed is outpacing memory access speed
• Ever changing architectures create a porting nightmare
Library use increases memory access cost

\[ q = Ap \]
\[ s = A^T r \]

```
DGEMV('n', m, n, alpha, a, lda, p, l, beta, q, l);
DGEMV('y', m, n, alpha, a, lda, r, l, beta, s, l);
```

```
for (i=0;i<n;++i)
  for (j=0;j<m;++j) { 
    q[j] += p[i]*A[i][j];
  }
for (i=0;i<n;++i)
  for (j=0;j<m;++j) { 
    s[i] += r[j]*A[i][j];
  }
```
Loop fusion is a powerful optimization

- This talk is about memory performance of loop fusion
- See also (among others)
  - Howell et al., 2008
  - Baker et al., 2006
  - Vuduc et al., 2003
  - Qasem, 2007
Build to Order Matrix Algebra

q = Ap
s = Aᵀr

for (i=0; i<N; ++i)
  for (j=0; j<M; ++j) {
    q[j] += p[i]*A[i][j];
    s[i] += r[j]*A[i][j];
  }
Subset of MATLAB

Generate Loops

Enumerate Fusion Opportunities

Analyze

Generate C Code
analytic model predicts runtimes for enumerated versions

Analytic Model

only best versions are tested

Empirical Testing

Code Generation

V1  V2  V3
Analytic model is based on reuse distances

RD(B) = 3

RD(A) = 5

Don’t end with from
Determine data location?
which determine where reads occur

RD(A) = 5
L1

L2
RD(B) = 3

Memory

Don’t end with from
Determine data location?
Model accurately predicts runtimes

\[ y = A A^T x \]

One version unless they stay difference
Model accurately predicts memory use

\[ y = A A^T x \]
Search Time

<table>
<thead>
<tr>
<th>Without Model</th>
<th>With Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiCGK</td>
<td></td>
</tr>
<tr>
<td>DGEMV</td>
<td></td>
</tr>
<tr>
<td>DGEMVT</td>
<td></td>
</tr>
<tr>
<td>AXPYDOT</td>
<td></td>
</tr>
<tr>
<td>WAXPBY</td>
<td></td>
</tr>
<tr>
<td>GEMVER</td>
<td></td>
</tr>
</tbody>
</table>

Seconds

- BiCGK
- DGEMV
- DGEMVT
- AXPYDOT
- WAXPBY
- GEMVER
But pushing to larger problems reveals trouble

DGEMV repeated nvecs times
in
  \( u_j \): vector,  \( A \): row matrix
out
  \( v_j \): vector
\{
  \( v_0 = A \times u_0 \)
  \( v_1 = A \times u_1 \)
  ...
  \( v_{nvecs-1} = A \times u_{nvecs-1} \)
\}
Defining Terms

for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
v0[i] += A[i][j] * u0[j]

for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
v1[i] += A[i][j] * u1[j]

for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
v0[i] += A[i][j] * u0[j]
v1[i] += A[i][j] * u1[j]

nvecs = 2

for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
v0[i] += A[i][j] * u0[j]

for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
v1[i] += A[i][j] * u1[j]

Figure: Three possible loop fusion options
An unfortunate crossover

\[ \text{nvecs} \times \text{matrix-vector multiply} \]
Not a cache problem

L1 Cache

L2 Cache
Or the TLB
It's the registers for the fused inner loops

<table>
<thead>
<tr>
<th>nvecs = 4</th>
<th>nvecs = 5</th>
<th>nvecs = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>faddl (%edi,%eax,8)</td>
<td>faddl (%esi,%eax,8)</td>
<td>mov 0x5c(%esp),%edi</td>
</tr>
<tr>
<td>fstpl (%edi,%eax,8)</td>
<td>fstpl (%esi,%eax,8)</td>
<td>faddl (%edi,%eax,8)</td>
</tr>
<tr>
<td>faddl (%esi,%eax,8)</td>
<td>mov 0x4c(%esp),%edi</td>
<td>faddl (%edi,%eax,8)</td>
</tr>
<tr>
<td>fstpl (%esi,%eax,8)</td>
<td>fstpl (%edi,%eax,8)</td>
<td>fstpl (%edi,%eax,8)</td>
</tr>
</tbody>
</table>
When we add registers to the model
Different trouble revealed by larger orders for fused outer loops
It’s the cache

Chart showing that its L2 cache performance that causes dropoff

- L2 misses per flop
- nvecs = 8

Legend:
- GotoBLAS
- all outer loops
- fully composed
What's the Best Amount of Fusion?

for
for
statement A
statement B
statement C
for
statement D
statement E
statement F

Issues when you get it wrong

Outer Loops:
cache misses

Inner Loops:
register spill
Questions

Special thanks:
Erik Silkensen

Contact:
jessup@cs.colorado.edu
ian.karlin@colorado.edu

http://ecee.colorado.edu/wpmu/btoblas/