Automated Empirical Optimizations of Software and the ATLAS project*

Software Engineering Seminar
Pascal Spörri

Is Search Really Necessary to Generate High-Performance BLAS?
Kamen Yotov and Xiaoming Li and Gang Ren and Maria Garzaran and David Padua and Keshav Pingali and Paul Stodghill
PROCEEDINGS OF THE IEEE, VOL. 93, NO. 2, FEBRUARY 2005
INTRODUCTION
BLAS (Basic Linear Algebra Subprograms)

• **Level 1**
  Vector operations
  \[ y \leftarrow \alpha x + z \]

• **Level 2**
  Matrix-Vector operations
  \[ y \leftarrow \alpha Ax + z \]

• **Level 3**
  Matrix-Matrix operations
  \[ D \leftarrow \alpha AB + \beta C \]
ATLAS
(Automatically Tuned Linear Algebra Software)

- Implements BLAS
- Applies empirical optimization techniques to source code to generate an optimized library
- Fully automatic
- Produces ANSI-C code
ARCHITECTURE
ATLAS Architecture

- Detect Hardware Parameters
- L1 Cache
- CPU parameters
- ATLAS Search Engine
- Parameters
- ATLAS Code Generator
- Multiple versions
- MFLOPS
- Source Code
- Execute And Measure

Kamen Yotov and Xiaoming Li and Gang Ren and Maria Garzaran and David Padua and Keshav Pingali and Paul Stodghill
PROCEEDINGS OF THE IEEE, VOL. 93, NO. 2, FEBRUARY 2005
ATLAS CODE GENERATOR
ATLAS Optimizations

- Case: Matrix-Matrix multiplication

\[
\begin{align*}
\text{for } i & \in [0:1:N - 1] \\
\text{for } j & \in [0:1:M - 1] \\
\text{for } k & \in [0:1:K - 1] \\
C_{i,j} &= C_{i,j} + A_{j,k} \times B_{k,j}
\end{align*}
\]
Loop Ordering

\[
\text{for } j \in [0:1:N - 1] \\
\text{for } i \in [0:1:M - 1] \\
\text{for } k \in [0:1:K - 1] \\
C_{i,j} = C_{i,j} + A_{j,k} \times B_{k,j}
\]
1st Level Blocking

For $i \in [0: N_B: N - 1]$
For $j \in [0: N_B: M - 1]$
For $k \in [0: N_B: K - 1]$

$N_B$ is chosen in such that the working set fits into $L_1$

$C_{i',j'} = C_{i,j'} + A_{j',k'} \times B_{k',j'}$

$$fもない^{for}$$
$$i \in [0: N_B: N - 1]$$
$$fもない^{for}$$
$$j \in [0: N_B: M - 1]$$
$$fもない^{for}$$
$$k \in [0: N_B: K - 1]$$

$K = M$
$N = M$
2nd Level Blocking

for i ∈ [0:NB:N − 1]
for j ∈ [0:NB:M − 1]
for k ∈ [0:NB:K − 1]

for j′ ∈ [j:NU:j + NB − 1]
for i′ ∈ [i:MU:i + NB − 1]
for k′ ∈ [k:KU:k + NB − 1]

for k'' ∈ [k':1:k' + KU − 1]
for j'' ∈ [j':1:j' + NU − 1]
for i'' ∈ [i':1:i' + MU − 1]

C_{i'',j''} = C_{i'',j''} + A_{j'',k''} \times B_{k'',j''}

M_U + N_U + M_U \times N_U \leq N_R

Unroll Loop

Graphic from “How To Write Fast Numerical Code: A Small Introduction”
Srinivas Chellappa, Franz Franchetti, and Markus Püschel
Scalar Replacement

• Replace array accesses with scalars

```c
double t[2];
for (i=0; i<8; i++) {
    t[0] = x[2*i] + x[2*i+1];
    t[1] = x[2*i] - x[2*i+1];
    y[2*i] = t[0] * D[2*i];
    y[2*i+1] = t[0] * D[2*i];
}
```

How To Write Fast Numerical Code: A Small Introduction
Srinivas Chellappa, Franz Franchetti, and Markus Püschel
Scalar Replacement

\[ a_{11} = A[1][1] \]
\[ a_{12} = A[1][2] \]
\[ a_{13} = A[1][3] \]
\[ a_{14} = A[1][4] \]
\[ \ldots \]
\[ b_{11} = B[1][1] \]
\[ b_{12} = B[1][2] \]
\[ b_{13} = B[1][3] \]
\[ b_{14} = B[1][4] \]
\[ \ldots \]
\[ c_{11} = a_{11} \times b_{11} \]
\[ c_{11} += a_{12} \times b_{21} \]
\[ c_{11} += a_{13} \times b_{31} \]
\[ \ldots \]
\[ c_{12} = a_{11} \times b_{12} \]
\[ c_{12} += a_{12} \times b_{22} \]
\[ c_{12} += a_{13} \times b_{32} \]
\[ \ldots \]
\[ c[1][1] = c_{11} \]
\[ c[1][2] = c_{12} \]
\[ c[1][3] = c_{13} \]
Data Hazards

LD  R1, 0(R2)
DSUB R4, R1, R5
AND R6, R1, R7
OR  R8, R1, R9
XOR R10, R1, R11

Skewing Factor
Pipeline Scheduling

Interleave \textit{mul} and \textit{add} sequences

\[\begin{align*}
\text{mul}_1 \\
\text{mul}_2 \\
\ldots \\
\text{mul}_{L_S} \\
\text{add}_1 \\
\text{mul}_{L_S+1} \\
\text{add}_2 \\
\text{mul}_{L_S+2} \\
\text{add}_3 \\
\ldots
\end{align*}\]

Skewing factor \(L_S\)
### Pipeline Scheduling

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a11 = A[1][1]</td>
<td>c11 = a11*b11</td>
</tr>
<tr>
<td>a12 = A[1][2]</td>
<td>c12 = a11*b12</td>
</tr>
<tr>
<td>a13 = A[1][3]</td>
<td>...</td>
</tr>
<tr>
<td>a14 = A[1][4]</td>
<td>...</td>
</tr>
<tr>
<td>b11 = B[1][1]</td>
<td>c11 += a12*b21</td>
</tr>
<tr>
<td>b12 = B[1][2]</td>
<td>c12 += a12*b22</td>
</tr>
<tr>
<td>b13 = B[1][3]</td>
<td>...</td>
</tr>
<tr>
<td>b14 = B[1][4]</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

C[1][1] = c11
C[1][2] = c12
C[1][3] = c13
EMPIRICAL OPTIMIZATION IN ATLAS
Detect Hardware Parameters → L1 Cache → ATLAS Search Engine → Parameters → ATLAS Code Generator → Source Code → Execute And Measure → Multiple versions

Optimize $f(x_1, x_2, x_3, \ldots, x_n)$

Kamen Yotov and Xiaoming Li and Gang Ren and Maria Garzaran and David Padua and Keshav Pingali and Paul Stodghill

PROCEEDINGS OF THE IEEE, VOL. 93, NO. 2, FEBRUARY 2005
Optimization Order

1. Find best block size for outer loop
2. Find best block sizes for inner loop
3. Find best skewing factor
4. Find best parameters for scheduling of loads
5. Additional parameters

\[ N \times M_U \times K = N_N \times N_R \]
Search for best Outer Loop Size

- $N_B$ must be a multiple of 4
- Use fastest version

Restrict search space $16 \leq N_B \leq \min(80, \sqrt{L_1 \text{Size}})$
DISCUSSION
Comparison to PhiPAC

**PhiPAC**
- Coding methodology to write fast code
- Precursor for ATLAS
- Specialized Code Generator for BLAS Matrix-Matrix Multiplication
- Optimizes parameters for inner and outer loop

**ATLAS**
- Library generator
- Automatic generation of optimized BLAS
- Support for handcoded routines
Comparison to eigen

matrix matrix product


Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz (x86_64)
Conclusion

Pro
• Fast method to generate an optimized library for a new platform
• Supports hand optimized code
• Implements BLAS

Contra
• Needs constant adjustment to support new architectures
• Outdated
Further Information

• ATLAS Project
  http://math-atlas.sourceforge.net/

• BLAS
  http://netlib.org/blas/