Automatic Performance Tuning and Machine Learning

Markus Püschel
Computer Science, ETH Zürich

with:
Frédéric de Mesmay
PhD, Electrical and Computer Engineering, Carnegie Mellon
PhD and Postdoc openings:

- High performance computing
- Compilers
- Theory
- Programming languages/Generative programming
Why Autotuning?

Matrix-Matrix Multiplication (MMM) on quadcore Intel platform
Performance [Gflop/s]

- Same (mathematical) operation count \(2n^3\)
- Compiler underperforms by 160x
Same for All Critical Compute Functions

DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)
Performance [Gflop/s]

WiFi Receiver (Physical layer) on one Intel Core
Throughput [Mbit/s] vs. Data rate [Mbit/s]
Solution: Autotuning

Definition: Search over alternative implementations or parameters to find the fastest.

Definition: Automating performance optimization with tools that complement/aid the compiler or programmer.

However: Search is an important tool. But expensive.

Solution: Machine learning
Organization

- Autotuning examples
- An example use of machine learning
time of implementation

platform known

time of installation

problem parameters known

time of use
Blocking improves locality

```c
double *c = calloc(sizeof(double), n*n);

void mmm(double *a, double *b, double *c, int n) {
    int i, j, k;
    for (i = 0; i < n; i+=B)
        for (j = 0; j < n; j+=B)
            for (k = 0; k < n; k+=B)
                for (il = i; il < i+B; il++)
                    for (j1 = j; j1 < j+B; j++)
                        for (kl = k; kl < k+B; kl++)
                            c[il*n+j1] += a[il*n + kl]*b[kl*n + j1];
}
```
PhiPac/ATLAS: MMM Generator

source: Pingali, Yotov, Cornell U.
ATLAS MMM generator

time of implementation

time of installation

platform known

time of use

problem parameters known
**Installation**

configure/make

**Usage**

\[ d = \text{dft}(n) \]

\[ d(x,y) \]

**Twiddles**

Search for fastest computation strategy

- **n = 1024**
  - radix 16
  - 16
    - base case
    - 8
      - base case
  - 64
    - radix 8
    - 8
      - base case
**FFTW: Codelet Generator**

*Frigo*

![Diagram]

- Input: `n`

- Output: `dft_n(*x, *y, ...)`

*fixed size DFT function*

*straightline code*
Carnegie Mellon
time of implementation

platform known

problem parameters known

FFTW codelet generator

ATLAS MMM generator

FFTW adaptive library

time of installation

time of use
OSKI: Sparse Matrix-Vector Multiplication

Vuduc, Im, Yelick, Demmel

- **Blocking for registers:**
  - Improves locality (reuse of input vector)
  - But creates overhead (zeros in block)
OSKI: Sparse Matrix-Vector Multiplication

Gain by blocking (dense MVM)

Overhead by blocking

\[
\frac{16}{9} = 1.77
\]

\[
\frac{1.4}{1.77} = 0.79 \text{ (no gain)}
\]
Carnegie Mellon

time of implementation
time of installation
platform known
problem parameters known

OSKI sparse MVM

ATLAS MMM generator

OSKI sparse MVM

FFTW codelet generator

OSKI parse MVM

FFTW adaptive library
Spiral: Linear Transforms & More

Algorithm knowledge

\[
\begin{align*}
\text{DFT}_n & \rightarrow P_{k/2, 2m}^T \left( \text{DFT}_{2m} \oplus \left( I_{k/2-1} \otimes i C_{2m} \text{rDFT}_{2m}(i/k) \right) \right) (\text{RDFT}_k \otimes I_m) \\
\text{rDFT}_{2n}(u) & \rightarrow L_{m}^k \left( I_k \otimes i \left( \text{rDFT}_{2m}\left( i + u / k \right) \right) \right) \left( \text{rDHT}_{2k}(u) \otimes I_m \right) \\
\text{RDFT-3}_n & \rightarrow (Q_{k/2, 2m} \otimes I_2) \left( I_k \otimes i \text{rDFT}_{2m}(i + 1/2/k) \right) (\text{RDFT-3}_k \otimes I_m)
\end{align*}
\]

Platform description

\[
\begin{align*}
A_m \otimes I_n & \rightarrow \left( L_{m}^{mp} \otimes I_{n/p} \right) \left( I_p \otimes (A_m \otimes I_{n/p}) \right) \left( L_{p}^{mp} \otimes I_{n/p} \right) \text{sm}(p, \mu) \\
I_m \otimes A_n & \rightarrow I_p \otimes (I_{m/p} \otimes A_n) \text{sm}(p, \mu) \\
(P \otimes I_n) & \rightarrow (P \otimes I_{n/\mu}) \text{sm}(p, \mu)
\end{align*}
\]

Optimized implementation

*regenerated for every new platform*
Program Generation in Spiral (Sketched)

Transform
user specified

Fast algorithm
in SPL
many choices

Σ-SPL

\[
\begin{align*}
\text{DFT}_8 & \\
(\text{DFT}_2 \otimes I_4) T_4^8 & (I_2 \otimes ((\text{DFT}_2 \otimes I_2) \\
& \cdot T_2^4 (I_2 \otimes \text{DFT}_2) L_2^4)) L_2^8 \\
\sum (S_j \text{DFT}_2 G_j) & \sum \left( \sum (S_{k,l} \text{diag}(t_{k,l}) \text{DFT}_2 G_{l}) \\
& \sum (S_m \text{diag}(t_m) \text{DFT}_2 G_{k,m}) \right) \\
\end{align*}
\]

Optimized implementation

Algorithm rules

Optimization at all abstraction levels

parallelization
vectorization
loop optimizations
constant folding
scheduling
......

+ search
Carnegie Mellon

time of implementation

Spiral: transforms general input size

FFTW codelet generator

Machine learning

Spiral: transforms fixed input size

OSKI sparse MVM

Machine learning

Spiral: transforms general input size

OSKI sparse MVM

Spiral: transforms general input size

ATLAS MMM generator

problem parameters known

time of use

time of installation

platform known

Machine learning

FFT W adaptive library

time of implementation
Organization

- Autotuning examples
- An example use of machine learning
Online tuning (time of use)

Installation

configure/make

Use

\( d = \text{dft}(n) \)
\( d(x,y) \)

Twiddles

Search for fastest computation strategy

Offline tuning (time of installation)

Installation

configure/make

for a few \( n \): search learn decision trees

Use

\( d = \text{dft}(n) \)
\( d(x,y) \)

Twiddles

Goal
Integration with Spiral-Generated Libraries

Voronenko 2008

\[(DFT_k \otimes I_m) \left( T_m^n(I_k \otimes DFT_m) L_k^n \right) + \text{some platform information} \]

\[
\begin{align*}
\text{DFT}_n & \rightarrow P_{k/2m}^T \left( DFT_{2m} \oplus \left( I_{k/2-1} \otimes C_{2m} \right) \right) \left( \text{RDFT}_k' \otimes I_m \right), \; k \text{ even}, \\
\text{RDFT}_n & \rightarrow \left( P_{k/2m}^T \otimes I_2 \right) \left( \text{RDFT}_{2m} \oplus \left( I_{k/2-1} \otimes D_{2m} \right) \right) \left( \text{RDFT}_k' \otimes I_m \right), \; k \text{ even}, \\
\text{DHT}_n & \rightarrow \left( P_{k/2m}^T \otimes I_2 \right) \left( DHT_{2m} \right) \left( \text{RDFT}_k' \otimes I_m \right), \; k \text{ even}, \\
\text{DHT}_n & \rightarrow \left( P_{k/2m}^T \otimes I_2 \right) \left( DHT_{2m} \right) \left( \text{RDFT}_k' \otimes I_m \right), \; k \text{ even},
\end{align*}
\]

\[
\begin{align*}
\text{RDFT-3}_n & \rightarrow (Q_{k/2m}^T \otimes I_2) \left( I_k \otimes I_2 \right) \left( \text{RDFT}_{3m} \right) \left( \text{RDFT}_k' \otimes I_m \right), \; k \text{ even}, \\
\text{DCT-2}_n & \rightarrow P_{k/2m}^T \left( \text{DCT-2}_m \right) \left( K_{2m}^T \otimes \left( I_{k/2-1} \otimes N_{2m} \right) \right) \left( \text{RDFT-3}_m \right) \left( B_n \left( t_{k/2}^n \otimes I_2 \right) \left( I_m \otimes \text{RDFT}_k' \right) \right) Q_{m/2,k}, \\
\text{DCT-3}_n & \rightarrow \text{DCT-2}_m, \\
\text{DCT-4}_n & \rightarrow Q_{k/2m}^T \left( I_k \otimes N_{2m} \right) \left( \text{RDFT-3}_m \right) \left( B_n' \left( t_{k/2}^n \otimes I_2 \right) \left( I_m \otimes \text{RDFT}_k \right) \right) Q_{m/2,k}, \\
\text{DFT}_n & \rightarrow \left( \text{DFT}_k \otimes I_m \right) \left( T_m^n \right) \left( I_k \otimes I_m \right) L_k^n, \; n = km, \; \text{gcd}(k,m) = 1 \\
\text{DFT}_p & \rightarrow P_n \left( \text{DFT}_k \otimes I_m \right) Q_n, \; n = km, \; \text{gcd}(k,m) = 1 \\
\text{DCT-3}_n & \rightarrow \left( I_m \otimes J_m \right) L_m^n \left( \text{DCT-3}_m(1/4) \oplus \text{DCT-3}_m(3/4) \right), \\
\text{DCT-4}_n & \rightarrow S_n \text{DCT-2}_n \text{diag}_{0 \leq k < n} \left( 1/(2 \cos((2k + 1)\pi/4n)) \right) \\
\text{IMDCT}_2m & \rightarrow \left( J_m \otimes I_m \otimes I_m \right) \left( \left[ \begin{array}{c} 1 \\ -1 \end{array} \right] \otimes I_m \right) J_{2m} \text{DCT-4}_{2m} \\
\text{WHT}_{2k} & \rightarrow \prod_{i=1}^{t} \left( I_{2^i} \otimes I_{2^{k-1}+\cdots+k_t} \right) \text{WHT}_{2^i} \otimes I_{2^{k-1}+\cdots+k_t}, \; k = k_1 + \cdots + k_t \\
\text{DFT}_2 & \rightarrow F_2 \\
\text{DCT-2}_2 & \rightarrow \text{diag}(1, 1/\sqrt{2}) F_2 \\
\text{DCT-4}_2 & \rightarrow J_2 R_{13\pi/8}
\end{align*}
\]
Organization

- Autotuning examples
- An example use of machine learning
  - Anatomy of an adaptive discrete Fourier transform library
  - Decision tree generation using C4.5
  - Results
Discrete/Fast Fourier Transform

- **Discrete Fourier transform (DFT):**
  \[ y = \text{DFT}_n x, \quad \text{DFT}_n = [e^{-2\pi i k\ell/n}]_{0 \leq k, \ell < n} \]

- **Cooley/Tukey fast Fourier transform (FFT):**
  \[ \text{DFT}_n = (\text{DFT}_k \otimes I_m) \top_m^n (I_k \otimes \text{DFT}_m) \odot_k^n, \quad n = km \]

- **Dataflow (right to left):** \( 16 = 4 \times 4 \)
Adaptive Scalar Implementation (FFTW 2.x)

```c
void dft(int n, cpx *y, cpx *x) {
    if (use_dft_base_case(n))
        dft_bc(n, y, x);
    else {
        int k = choose_dft_radix(n);
        for (int i=0; i < k; ++i)
            dft_strided(m, k, t + m*i, x + m*i);
        for (int i=0; i < m; ++i)
            dft_scaled(k, m, precomp_d[i], y + i, t + i);
    }
}

void dft_strided(int n, int istr, cpx *y, cpx *x) { ... }
void dft_scaled(int n, int str, cpx *d, cpx *y, cpx *x) { ... }
```

Choices used for adaptation
void dft(int n, cpx *y, cpx *x) {
    if (use_dft_base_case(n))
        dft_bc(n, y, x);
    else {
        int k = choose_dft_radix(n);
        for (int i=0; i < k; ++i)
            dft_strided(m, k, t + m*i, x + m*i);
        for (int i=0; i < m; ++i)
            dft_scaled(k, m, precomp_d[i], y + i, t + i);
    }
}
void dft_strided(int n, int istr, cpx *y, cpx *x) { ... }
void dft_scaled(int n, int str, cpx *d, cpx *y, cpx *x) { ... }

Decision Graph of Library

Choices used for adaptation
Spiral-Generated Libraries

\[(\text{DFT}_k \otimes \text{I}_m) \sqcap^n_m (\text{I}_k \otimes \text{DFT}_m) \sqcup^n_k\]

- **Spiral**
  - Standard Scalar
  - Vectorized
  - Threading
  - Buffering

- 20 mutually recursive functions
- 10 different choices (occurring recursively)
- Choices are heterogeneous (radix, threading, buffering, ...)

OpenMP loop of scaled dfts
Upon installation, generate decision trees for each choice

Example:

```c
if ( n <= 65536 ) {
    if ( n <= 32 ) {
        if ( n <= 4 ) {return 2;}
        else {return 4;}
    }
    else {
        if ( n <= 1024 ) {
            if ( n <= 256 ) {return 8;}
            else {return 32;}
        }
        else {
            .................
        }
    }
}
```
Statistical Classification: C4.5

Features (events)

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>don’t play</td>
</tr>
<tr>
<td>sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>don’t play</td>
</tr>
<tr>
<td>overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>play</td>
</tr>
<tr>
<td>rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>play</td>
</tr>
<tr>
<td>rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>play</td>
</tr>
<tr>
<td>rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>don’t play</td>
</tr>
<tr>
<td>overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>play</td>
</tr>
<tr>
<td>sunny</td>
<td>72</td>
<td>95</td>
<td>false</td>
<td>don’t play</td>
</tr>
<tr>
<td>sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>play</td>
</tr>
<tr>
<td>rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>play</td>
</tr>
<tr>
<td>sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>play</td>
</tr>
<tr>
<td>overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>play</td>
</tr>
<tr>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>play</td>
</tr>
<tr>
<td>rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>don’t play</td>
</tr>
</tbody>
</table>

P(play|windy=false) = 6/8
P(don’t play|windy=false) = 2/8
P(play|windy=true) = 1/2
P(don’t play|windy=false) = 1/2

H(windy=false) = 0.81
H(windy=true) = 1.0

Entropy of Features

H(windy) = 0.89
H(outlook) = 0.69
H(humidity) = ...
Application to Libraries

- Features = arguments of functions (except variable pointers)

\[
\text{dft}(\textint n, \text{cpx} \ *y, \text{cpx} \ *x)
\]
\[
\text{dft\_strided}(\textint n, \textint istr, \text{cpx} \ *y, \text{cpx} \ *x)
\]
\[
\text{dft\_scaled}(\textint n, \textint str, \text{cpx} \ *d, \text{cpx} \ *y, \text{cpx} \ *x)
\]

- At installation time:
  - Run search for a few input sizes \( n \)
  - Yields training set: features and associated decisions (several for each size)
  - Generate decision trees using C4.5 and insert into library
Issues

- Correctness of generated decision trees
  - Issue: learning sizes $n$ in $\{12, 18, 24, 48\}$, may find radix 6
  - Solution: correction pass through decision tree

- Prime factor structure

  \[ n = 2^i 3^j = 2, 3, 4, 6, 9, 12, 16, 18, 24, 27, 32, \ldots \]

  Compute $i, j$ and add to features
Experimental Setup

- 3GHz Intel Xeon 5160 (2 Core 2 Duos = 4 cores)
- Linux 64-bit, icc 10.1
- Libraries:
  - IPP 5.3
  - FFTW 3.2 alpha 2
  - Spiral-generated library
Learning works as expected
“All” Sizes

Complex DFT, double precision, mixed sizes
Performance [GFlop/s]

- All sizes $n \leq 2^{18}$, with prime factors $\leq 19$
“All” Sizes

- All sizes $n \leq 2^{18}$, with prime factors $\leq 19$
- Higher order fit of all sizes
Related Work

- **Machine learning in Spiral**
  - Learning DFT recursions (Singer/Veloso 2001)

- **Machine learning in compilation**
  - Scheduling (Moss et al. 1997, Cavazos/Moss 2004)
  - Branch prediction (Calder et al. 1997)
  - Heuristics generation (Monsifrot/Bodin/Quiniou 2002)
  - Feature generation (Leather/Bonilla/O’Boyle 2009)
  - Multicores (Wang/O’Boyle 2009)
This talk

- Frédéric de Mesmay, Yevgen Voronenko and Markus Püschel
  Offline Library Adaptation Using Automatically Generated Heuristics
  Proc. International Parallel and Distributed Processing Symposium (IPDPS), pp. 1-10, 2010

- Frédéric de Mesmay, Arpad Rimmel, Yevgen Voronenko and Markus Püschel
  Bandit-Based Optimization on Graphs with Application to Library Performance Tuning
Message of Talk

- **Machine learning should be used in autotuning**
  - Overcomes the problem of expensive searches
  - Relatively easy to do
  - Applicable to any search-based approach