Automatic Performance Tuning and Machine Learning

Markus Püschel
Computer Science, ETH Zürich

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

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

www.spiral.net

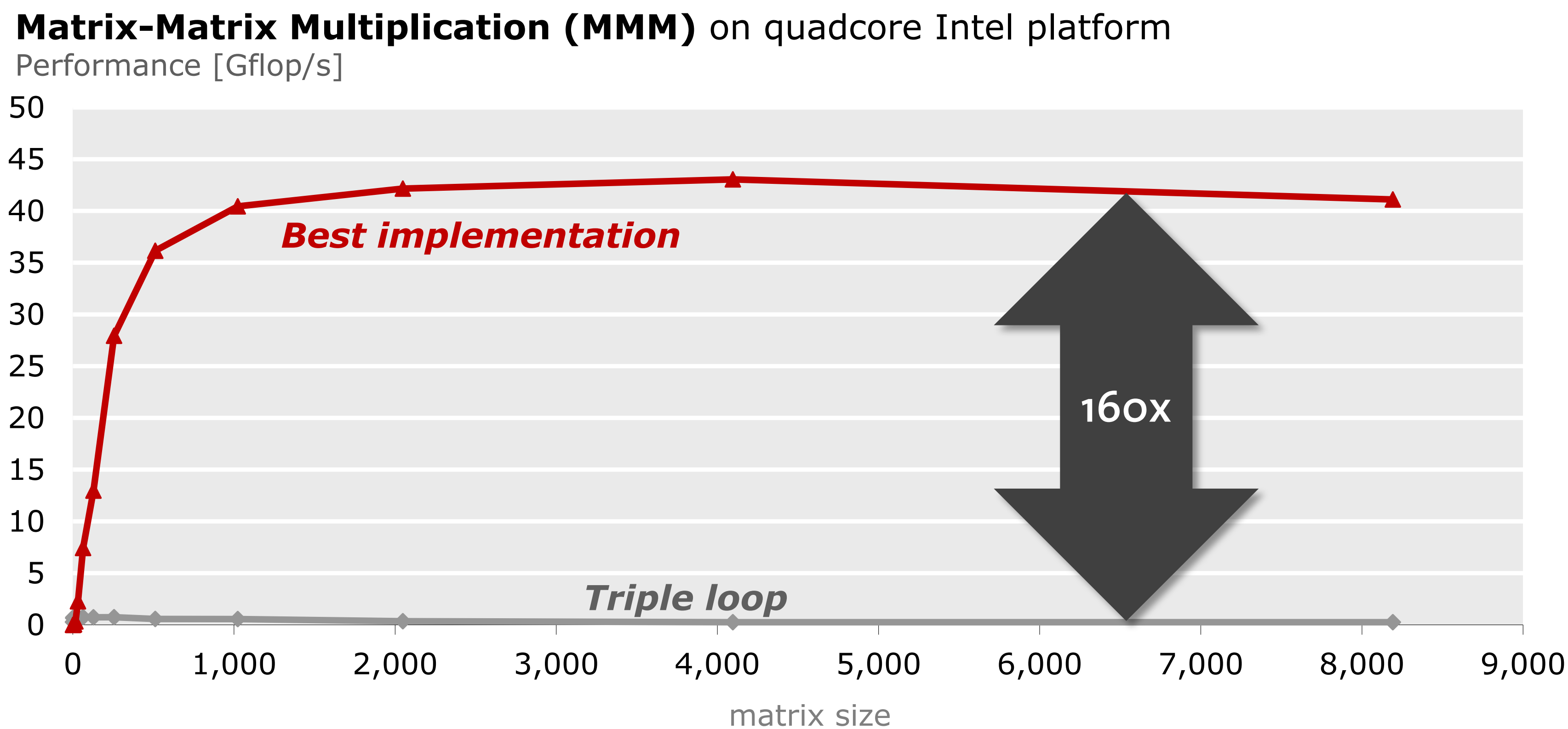
ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

PhD and Postdoc openings:

- High performance computing
- Compilers
- Theory
- Programming languages/Generative programming

Why Autotuning?

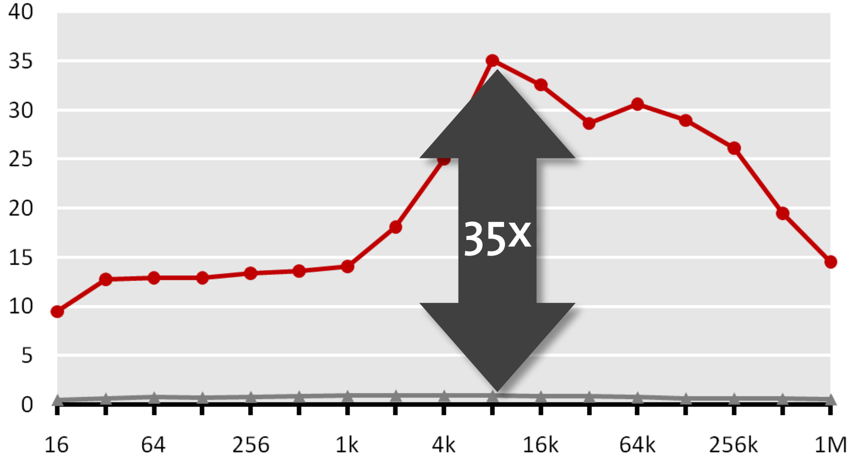


- Same (mathematical) operation count ($2n^3$)
- Compiler underperforms by 160x

Same for All Critical Compute Functions

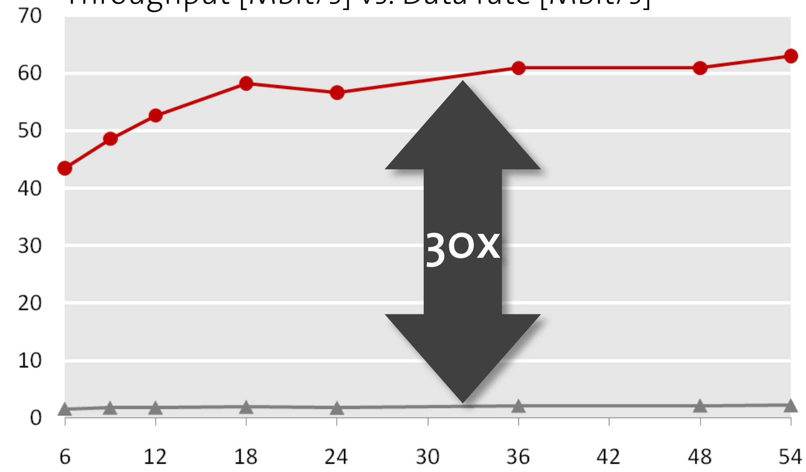
DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)

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.*~~

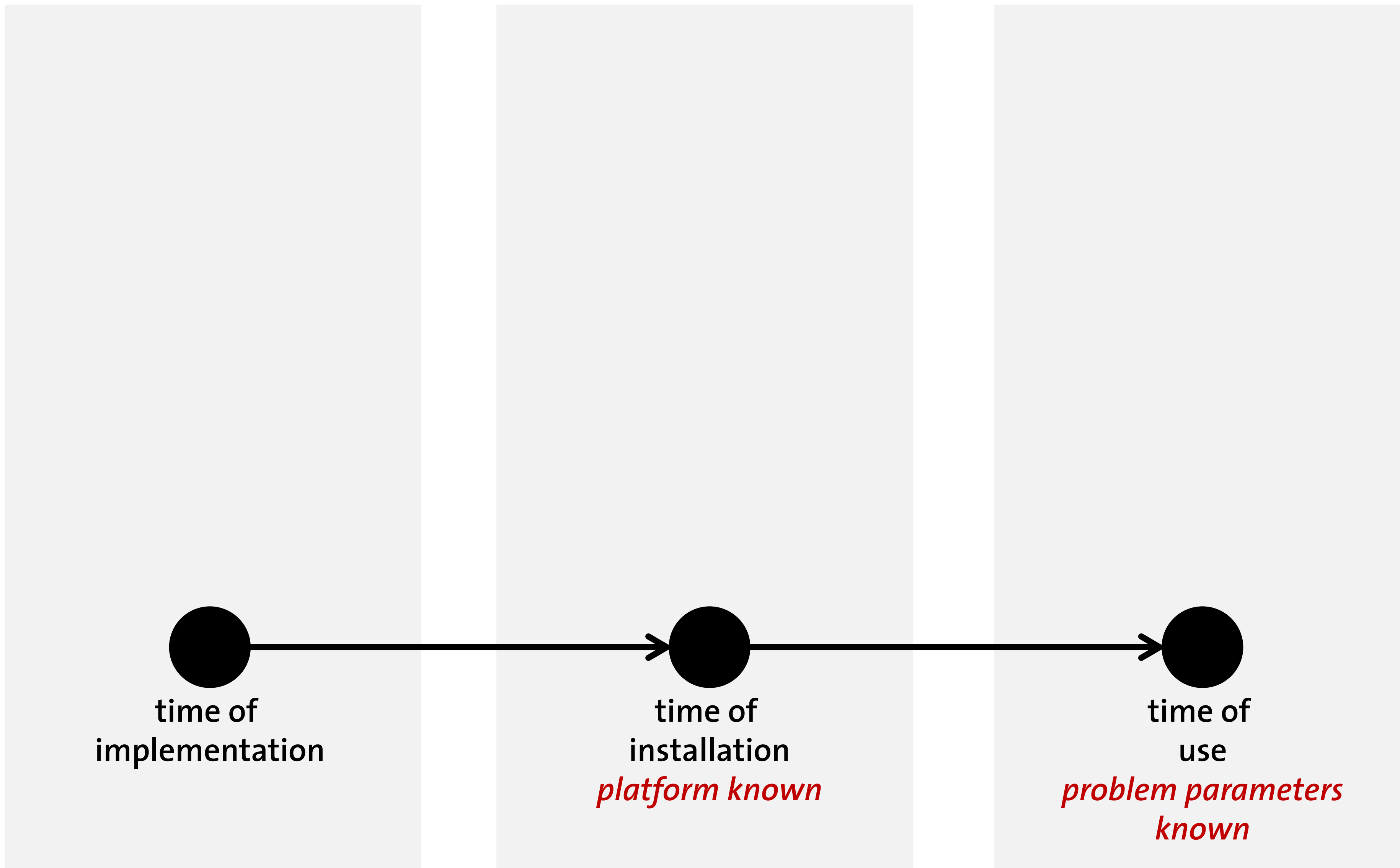
Definition: *Automating performance optimization with tools that complement/aid the compiler or programmer.*

However: *Search is an important tool. But expensive.*

Solution: Machine learning

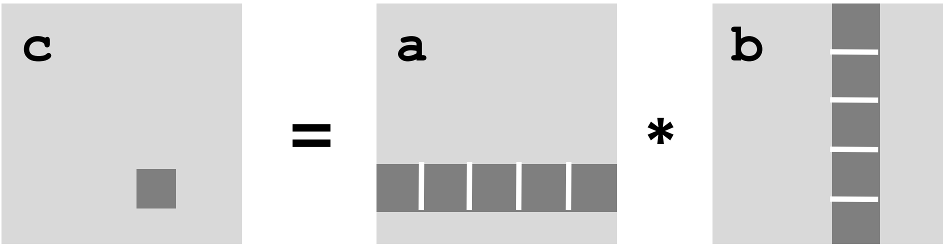
Organization

- Autotuning examples
- An example use of machine learning


time of
implementation
time of
installation
platform known
time of
use
problem parameters
known

PhiPac/ATLAS: MMM Generator

Whaley, Bilmes, Demmel, Dongarra, ...

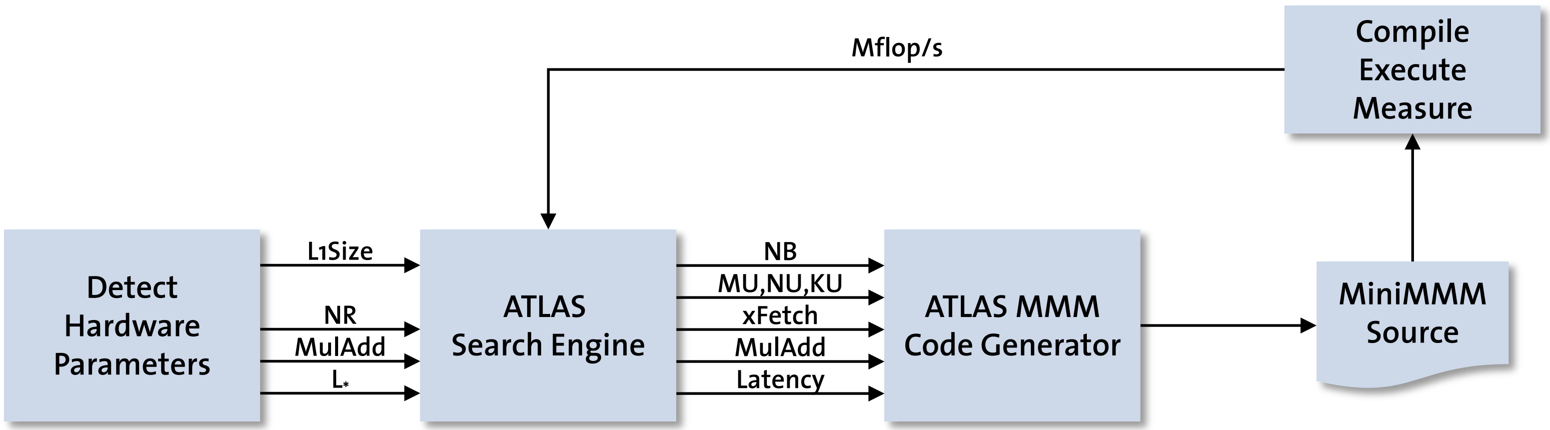


Blocking improves locality

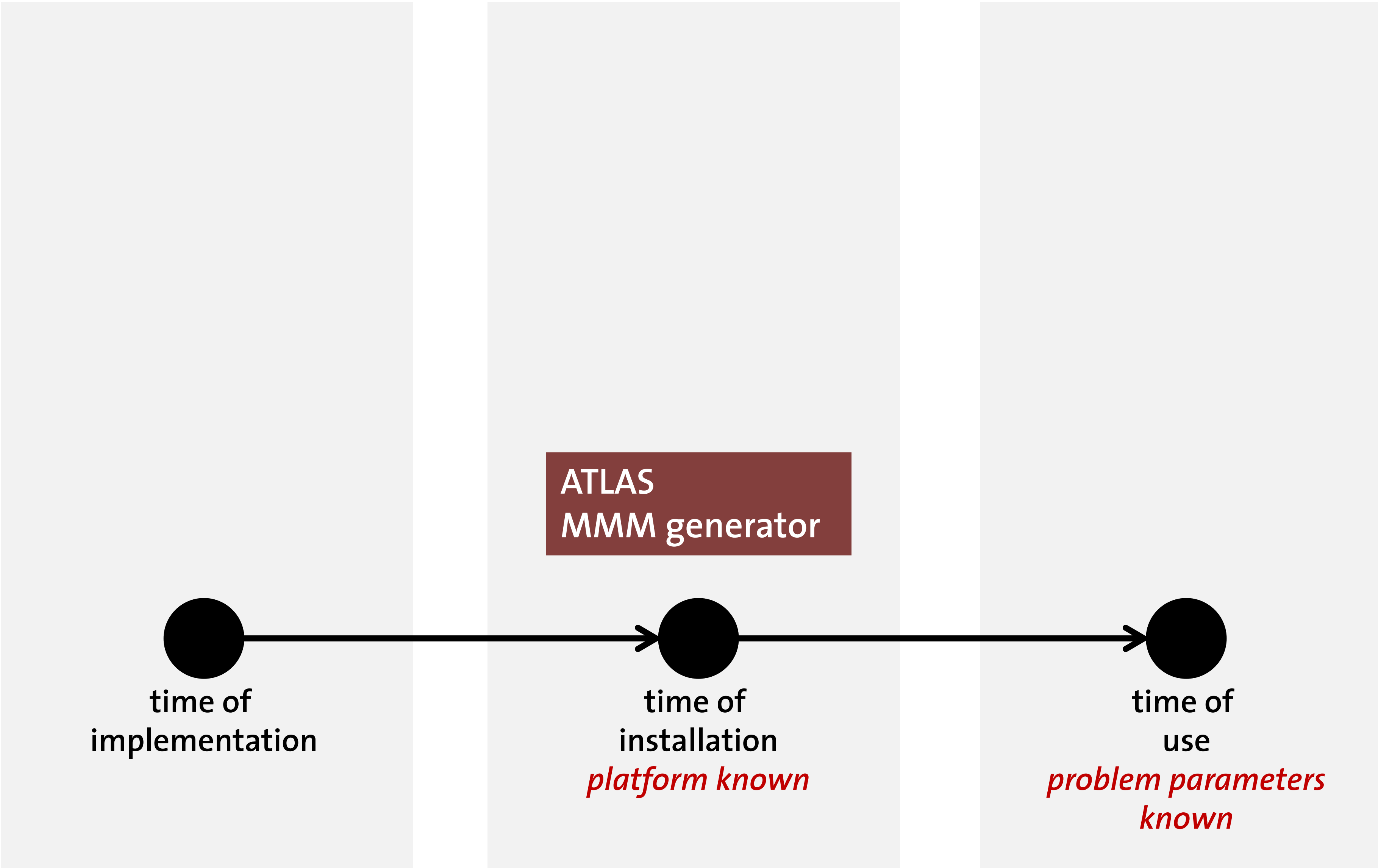
```
c = (double *) calloc(sizeof(double), n*n);

/* Multiply n x n matrices a and b  */
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)
                    /* B x B mini matrix multiplications */
                    for (i1 = i; i1 < i+B; i++)
                        for (j1 = j; j1 < j+B; j++)
                            for (k1 = k; k1 < k+B; k++)
                                 c[i1*n+j1] += a[i1*n + k1]*b[k1*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

FFTW: Discrete Fourier Transform (DFT)

Frigo, Johnson

Installation

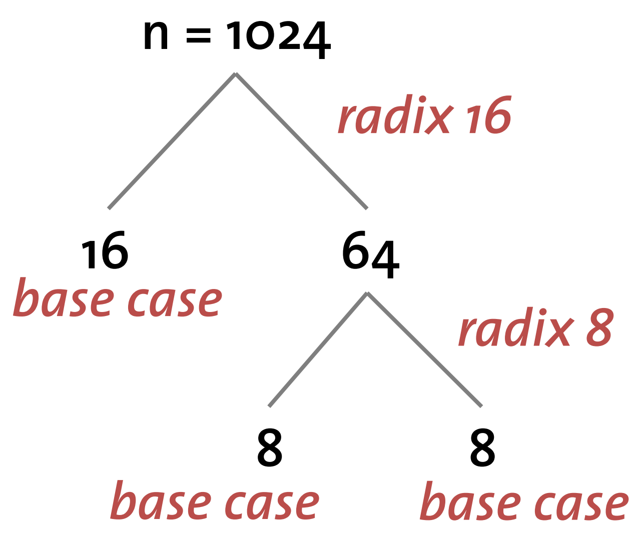
```
configure/make
```

Usage

```
d = dft(n)
d(x,y)
```

Twiddles

Search for fastest computation strategy



FFTW: Codelet Generator

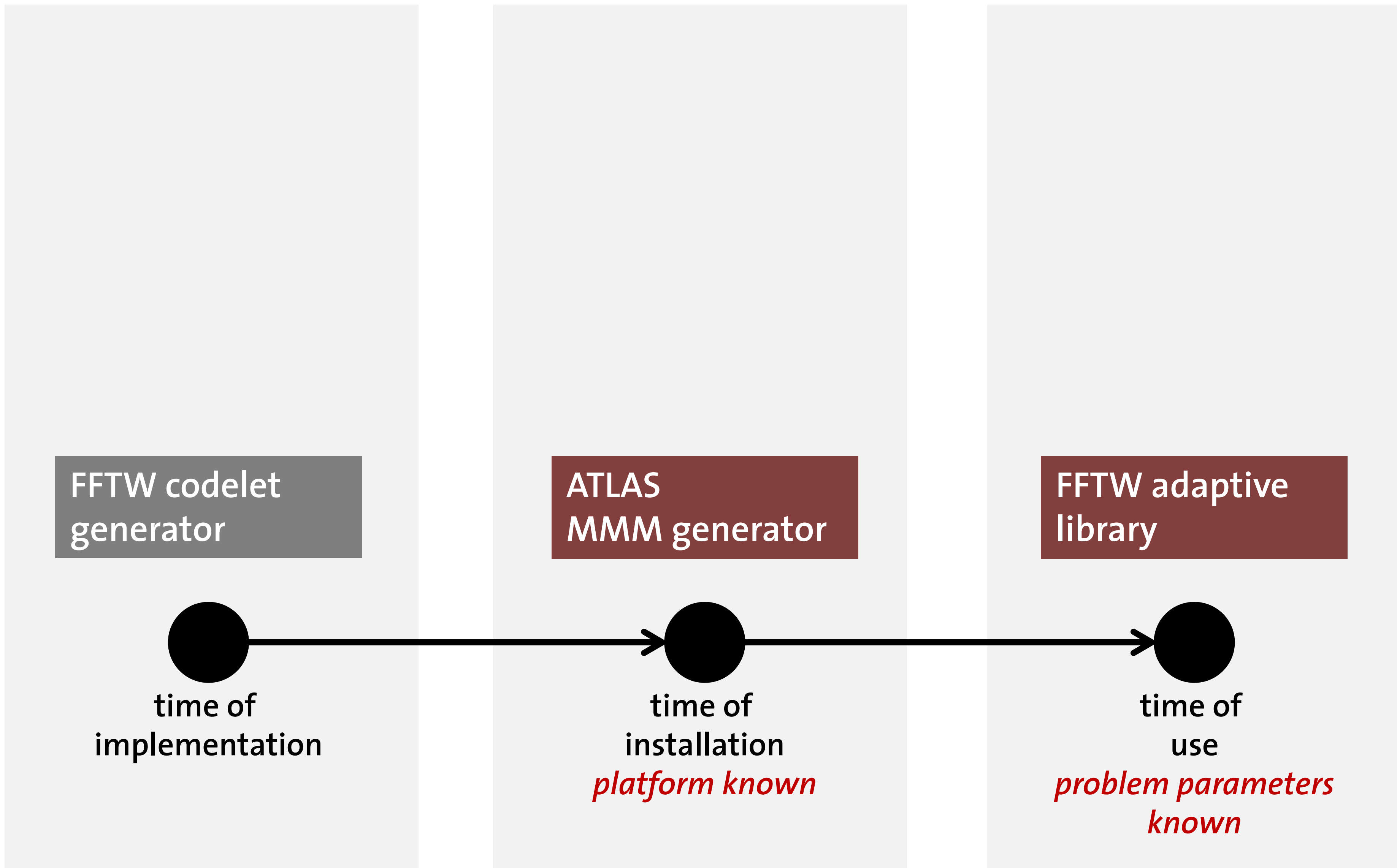
Frigo

n

DFT codelet generator

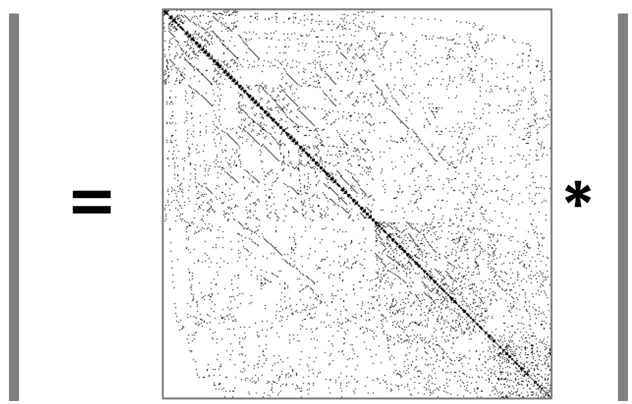
```
dft_n(*x, *y, …)
```

fixed size DFT function
straightline code

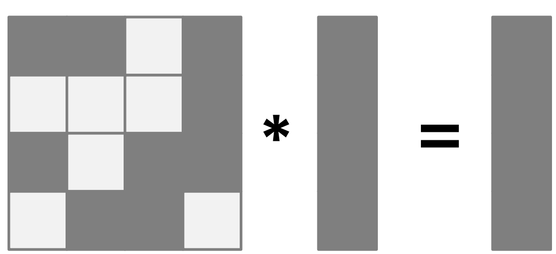

FFTW codelet generator
ATLAS MMM generator
FFTW adaptive library
time of implementation
time of installation
platform known
time of use
problem parameters known

OSKI: Sparse Matrix-Vector Multiplication

Vuduc, Im, Yelick, Demmel

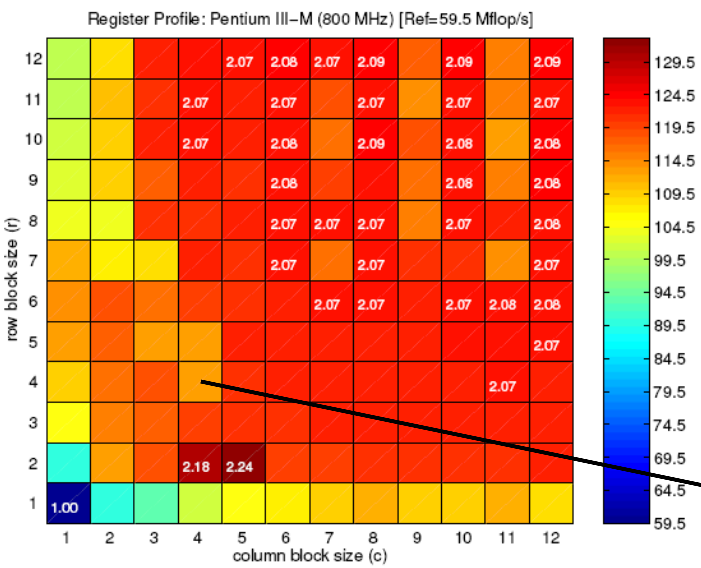


- **Blocking for registers:**
  - Improves locality (reuse of input vector)
  - But creates overhead (zeros in block)



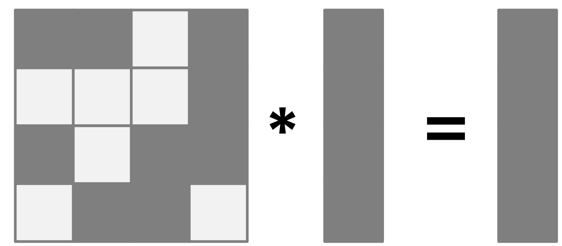
OSKI: Sparse Matrix-Vector Multiplication

Gain by blocking (dense MVM)



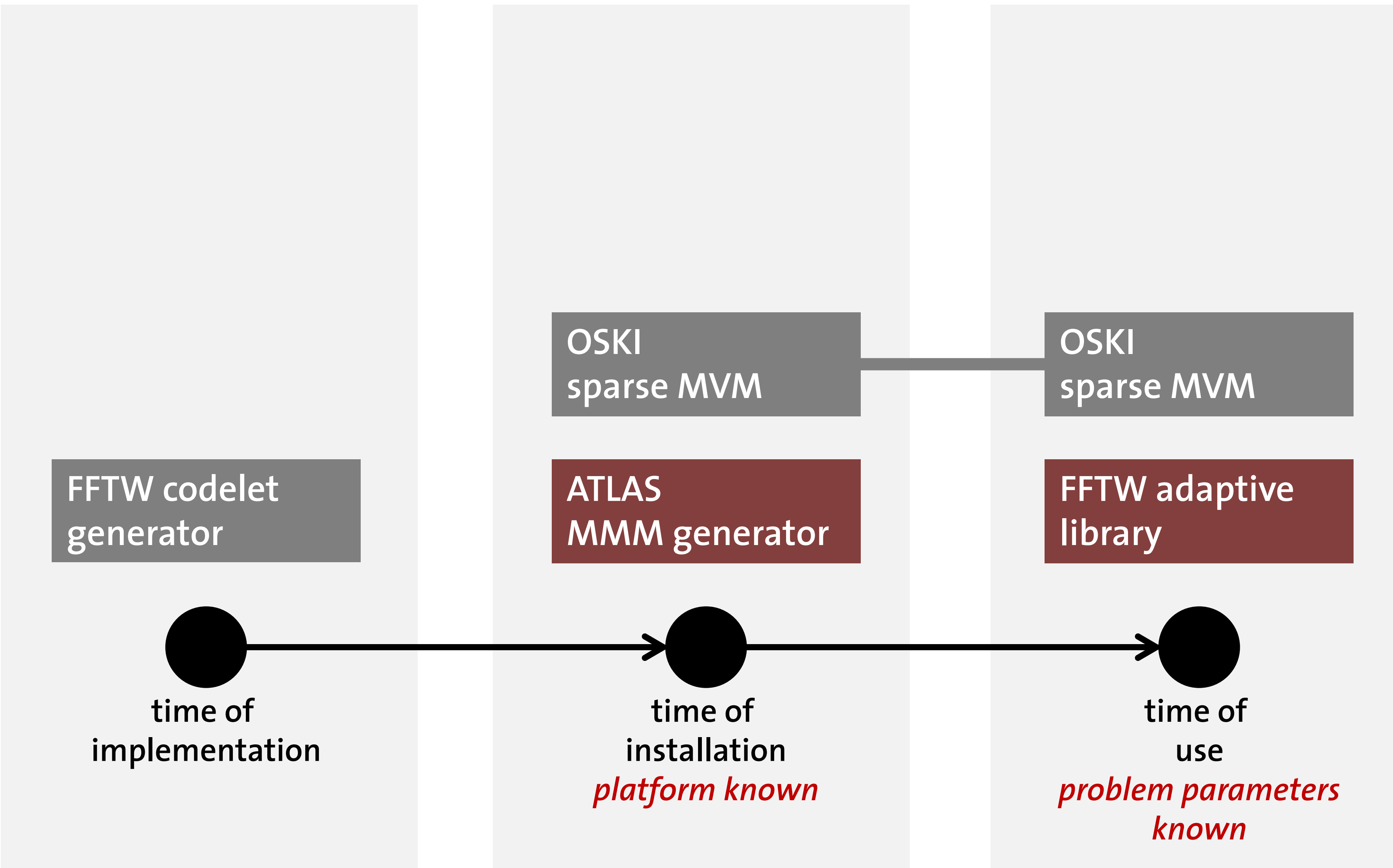
1.4

Overhead by blocking



16/9 = 1.77

1.4/1.77 = 0.79 (no gain)


OSKI
sparse MVM
OSKI
sparse MVM
FFTW codelet
generator
ATLAS
MMM generator
FFTW adaptive
library
time of
implementation
time of
installation
platform known
time of
use
problem parameters
known

Spiral: Linear Transforms & More

Algorithm knowledge

$$\mathbf{DFT}_n \to P_{k/2,2m}^{\top}\left(\mathbf{DFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i C_{2m}\,\mathbf{rDFT}_{2m}(i/k)\right)\right)\left(\mathbf{RDFT}'_k \otimes I_m\right)$$

$$\begin{vmatrix}\mathbf{rDFT}_{2n}(u)\\ \mathbf{rDHT}_{2n}(u)\end{vmatrix} \to L_m^{2n}\left(I_k \otimes_i \begin{vmatrix}\mathbf{rDFT}_{2m}((i+u)/k)\\ \mathbf{rDHT}_{2m}((i+u)/k)\end{vmatrix}\right)\left(\begin{vmatrix}\mathbf{rDFT}_{2k}(u)\\ \mathbf{rDHT}_{2k}(u)\end{vmatrix} \otimes I_m\right)$$

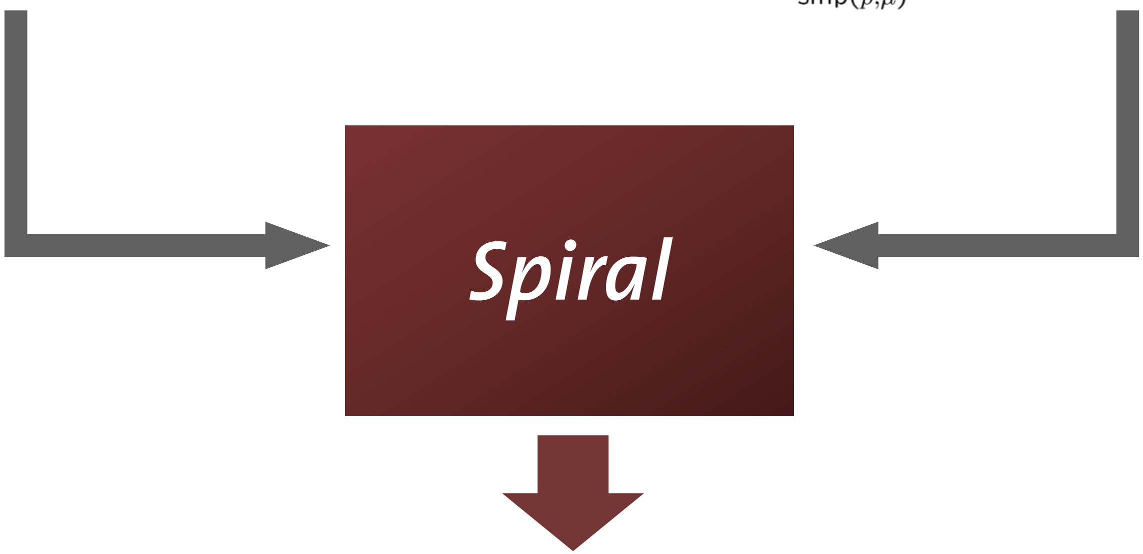
$$\mathbf{RDFT\text{-}3}_n \to (Q_{k/2,m}^{\top} \otimes I_2)\,(I_k \otimes_i \mathbf{rDFT}_{2m})(i+1/2)/k))\,(\mathbf{RDFT\text{-}3}_k \otimes I_m)$$

Platform description

$$\underbrace{A_m \otimes \mathrm{I}_n}_{\mathsf{smp}(p,\mu)} \to \underbrace{\left(\mathsf{L}_m^{mp} \otimes \mathrm{I}_{n/p}\right)\left(\mathrm{I}_p \otimes (A_m \otimes \mathrm{I}_{n/p})\right)\left(\mathsf{L}_p^{mp} \otimes \mathrm{I}_{n/p}\right)}_{\mathsf{smp}(p,\mu)}$$

$$\underbrace{\mathrm{I}_m \otimes A_n}_{\mathsf{smp}(p,\mu)} \to \mathrm{I}_p \otimes_{\parallel}\left(\mathrm{I}_{m/p} \otimes A_n\right)$$

$$\underbrace{(P \otimes \mathrm{I}_n)}_{\mathsf{smp}(p,\mu)} \to \left(P \otimes \mathrm{I}_{n/\mu}\right)\overline{\otimes}\,\mathrm{I}_\mu$$



Optimized implementation

regenerated for every new platform

Program Generation in Spiral (Sketched)

Transform
user specified

$$\mathrm{DFT}_8$$

Algorithm rules

Fast algorithm in SPL
many choices

$$(\mathrm{DFT}_2 \otimes I_4)\, T^8_4\, (I_2 \otimes ((\mathrm{DFT}_2 \otimes I_2) \cdot T^4_2\, (I_2 \otimes \mathrm{DFT}_2)\, L^4_2))\, L^8_2$$

Σ-SPL

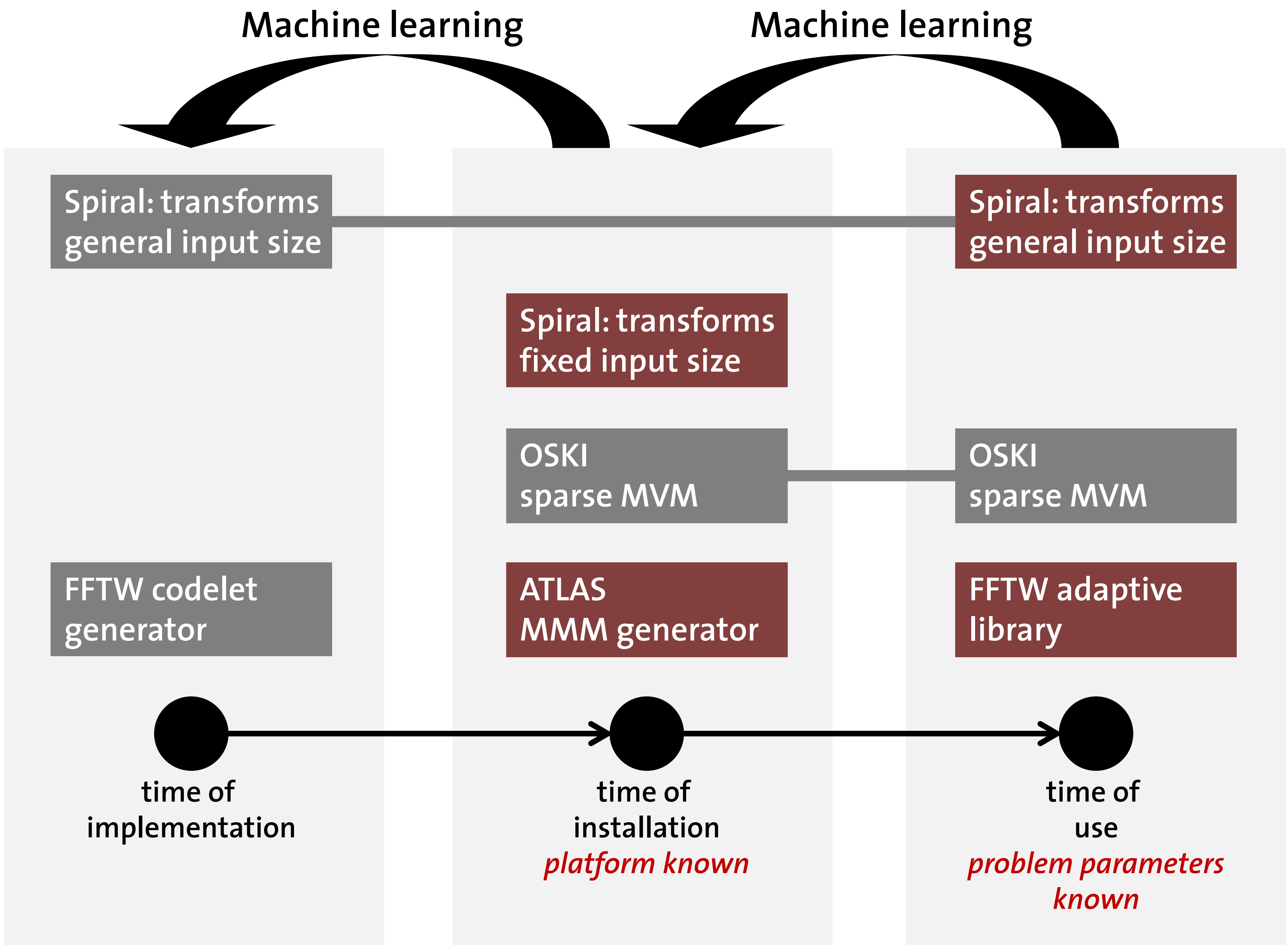
$$\sum \left(S_j\, \mathrm{DFT}_2\, G_j\right) \sum \left(\sum \left(S_{k,l}\, \mathrm{diag}\left(t_{k,l}\right) \mathrm{DFT}_2\, G_l\right) \sum \left(S_m\, \mathrm{diag}\left(t_m\right) \mathrm{DFT}_2\, G_{k,m}\right)\right)$$

Optimized implementation

Optimization at all abstraction levels

- parallelization
  vectorization
- loop
  optimizations
- constant folding
  scheduling
  ……

+ search



Machine learning
Machine learning
Spiral: transforms general input size
Spiral: transforms general input size
Spiral: transforms fixed input size
OSKI sparse MVM
OSKI sparse MVM
FFTW codelet generator
ATLAS MMM generator
FFTW adaptive library
time of implementation
time of installation
platform known
time of use
problem parameters known

Organization

- Autotuning examples
- An example use of machine learning

Online tuning
(time of use)

Installation

```
configure/make
```

Use

```
d = 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 = dft(n)
d(x,y)
```

Twiddles

Goal

Integration with Spiral-Generated Libraries

Voronenko 2008

$(\mathbf{DFT}_k \otimes \mathrm{I}_m)\,\mathrm{T}^n_m(\mathrm{I}_k \otimes \mathbf{DFT}_m)\,\mathrm{L}^n_k$
+ some platform information → **Spiral** → **Online tunable library**

$$\mathbf{DFT}_n \to P^\top_{k/2,2m}\left(\mathbf{DFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i C_{2m}\,\mathbf{rDFT}_{2m}(i/k)\right)\right)\left(\mathbf{RDFT}'_k \otimes I_m\right), \quad k \text{ even},$$

$$\left|\begin{matrix}\mathbf{RDFT}_n\\ \mathbf{RDFT}'_n\\ \mathbf{DHT}_n\\ \mathbf{DHT}'_n\end{matrix}\right| \to (P^\top_{k/2,m} \otimes I_2)\left(\left|\begin{matrix}\mathbf{RDFT}_{2m}\\ \mathbf{RDFT}'_{2m}\\ \mathbf{DHT}_{2m}\\ \mathbf{DHT}'_{2m}\end{matrix}\right| \oplus \left(I_{k/2-1} \otimes_i D_{2m}\left|\begin{matrix}\mathbf{rDFT}_{2m}(i/k)\\ \mathbf{rDFT}_{2m}(i/k)\\ \mathbf{rDHT}_{2m}(i/k)\\ \mathbf{rDHT}_{2m}(i/k)\end{matrix}\right|\right)\right)\left(\left|\begin{matrix}\mathbf{RDFT}'_k\\ \mathbf{RDFT}'_k\\ \mathbf{DHT}'_k\\ \mathbf{DHT}'_k\end{matrix}\right| \otimes I_m\right), \quad k \text{ even},$$

$$\left|\begin{matrix}\mathbf{rDFT}_{2n}(u)\\ \mathbf{rDHT}_{2n}(u)\end{matrix}\right| \to L^{2n}_m\left(I_k \otimes_i \left|\begin{matrix}\mathbf{rDFT}_{2m}((i+u)/k)\\ \mathbf{rDHT}_{2m}((i+u)/k)\end{matrix}\right|\right)\left(\left|\begin{matrix}\mathbf{rDFT}_{2k}(u)\\ \mathbf{rDHT}_{2k}(u)\end{matrix}\right| \otimes I_m\right),$$

$$\mathbf{RDFT\text{-}3}_n \to (Q^\top_{k/2,m} \otimes I_2)\,(I_k \otimes_i \mathbf{rDFT}_{2m})(i+1/2)/k))\,(\mathbf{RDFT\text{-}3}_k \otimes I_m), \quad k \text{ even},$$

$$\mathbf{DCT\text{-}2}_n \to P^\top_{k/2,2m}\left(\mathbf{DCT\text{-}2}_{2m}\,K^{2m}_2 \oplus \left(I_{k/2-1} \otimes N_{2m}\,\mathbf{RDFT\text{-}3}^\top_{2m}\right)\right) B_n(L^{n/2}_{k/2} \otimes I_2)(I_m \otimes \mathbf{RDFT}'_k)Q_{m/2,k},$$

$$\mathbf{DCT\text{-}3}_n \to \mathbf{DCT\text{-}2}^\top_n,$$

$$\mathbf{DCT\text{-}4}_n \to Q^\top_{k/2,2m}\left(I_{k/2} \otimes N_{2m}\,\mathbf{RDFT\text{-}3}^\top_{2m}\right) B'_n(L^{n/2}_{k/2} \otimes I_2)(I_m \otimes \mathbf{RDFT\text{-}3}_k)Q_{m/2,k}.$$

$$\begin{aligned}
\mathbf{DFT}_n &\to (\mathbf{DFT}_k \otimes \mathrm{I}_m)\,\mathrm{T}^n_m(\mathrm{I}_k \otimes \mathbf{DFT}_m)\,\mathrm{L}^n_k, \quad n = km\\
\mathbf{DFT}_n &\to P_n(\mathbf{DFT}_k \otimes \mathbf{DFT}_m)Q_n, \quad n = km,\ \gcd(k,m) = 1\\
\mathbf{DFT}_p &\to R^T_p(\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1})D_p(\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1})R_p, \quad p \text{ prime}\\
\mathbf{DCT\text{-}3}_n &\to (\mathrm{I}_m \oplus \mathrm{J}_m)\,\mathrm{L}^n_m(\mathbf{DCT\text{-}3}_m(1/4) \oplus \mathbf{DCT\text{-}3}_m(3/4))\\
&\qquad \cdot(\mathrm{F}_2 \otimes \mathrm{I}_m)\begin{bmatrix}\mathrm{I}_m & 0 \oplus -\mathrm{J}_{m-1}\\ & \frac{1}{\sqrt{2}}(\mathrm{I}_1 \oplus 2\,\mathrm{I}_m)\end{bmatrix}, \quad n = 2m\\
\mathbf{DCT\text{-}4}_n &\to S_n\mathbf{DCT\text{-}2}_n\,\mathrm{diag}_{0\le k<n}(1/(2\cos((2k+1)\pi/4n)))\\
\mathbf{IMDCT}_{2m} &\to (\mathrm{J}_m \oplus \mathrm{I}_m \oplus \mathrm{I}_m \oplus \mathrm{J}_m)\left(\left(\begin{bmatrix}1\\-1\end{bmatrix} \otimes \mathrm{I}_m\right) \oplus \left(\begin{bmatrix}-1\\-1\end{bmatrix} \otimes \mathrm{I}_m\right)\right)\mathrm{J}_{2m}\,\mathbf{DCT\text{-}4}_{2m}\\
\mathbf{WHT}_{2^k} &\to \prod_{i=1}^{t}(\mathrm{I}_{2^{k_1+\cdots+k_{i-1}}} \otimes \mathbf{WHT}_{2^{k_i}} \otimes \mathrm{I}_{2^{k_{i+1}+\cdots+k_t}}), \quad k = k_1 + \cdots + k_t\\
\mathbf{DFT}_2 &\to \mathrm{F}_2\\
\mathbf{DCT\text{-}2}_2 &\to \mathrm{diag}(1, 1/\sqrt{2})\,\mathrm{F}_2\\
\mathbf{DCT\text{-}4}_2 &\to \mathrm{J}_2\,\mathrm{R}_{13\pi/8}
\end{aligned}$$

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

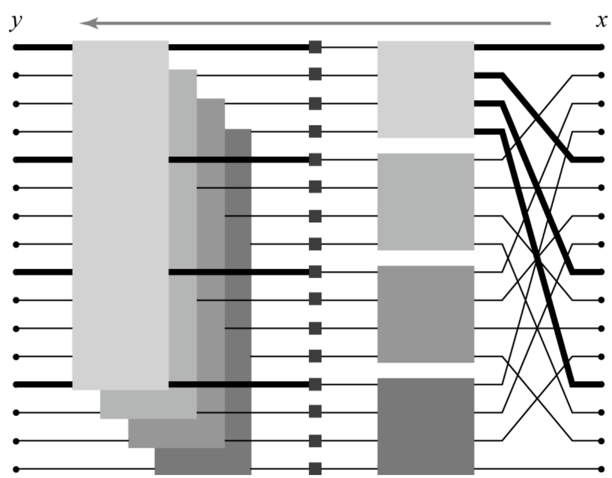
- Discrete Fourier transform (DFT):

$$y = \mathbf{DFT}_n\, x, \qquad \mathbf{DFT}_n = [e^{-2k\ell\pi i/n}]_{0\le k,\ell<n}$$

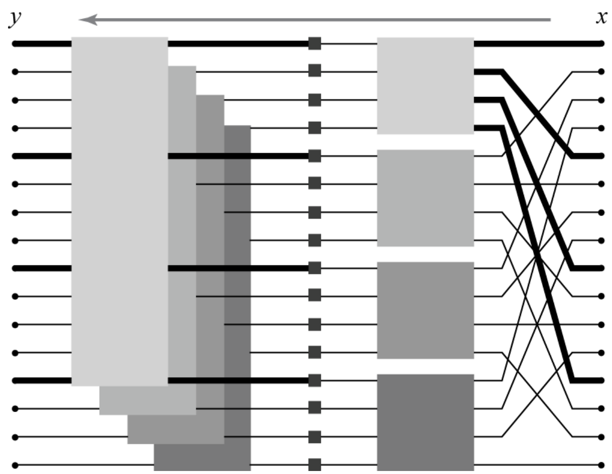
- Cooley/Tukey fast Fourier transform (FFT):

$$\mathbf{DFT}_n = (\mathbf{DFT}_k \otimes \mathrm{I}_m)\, \mathsf{T}^n_m (\mathrm{I}_k \otimes \mathbf{DFT}_m)\, \mathsf{L}^n_k, \quad n = km$$

- Dataflow (right to left): 16 = 4 x 4



Adaptive Scalar Implementation (FFTW 2.x)



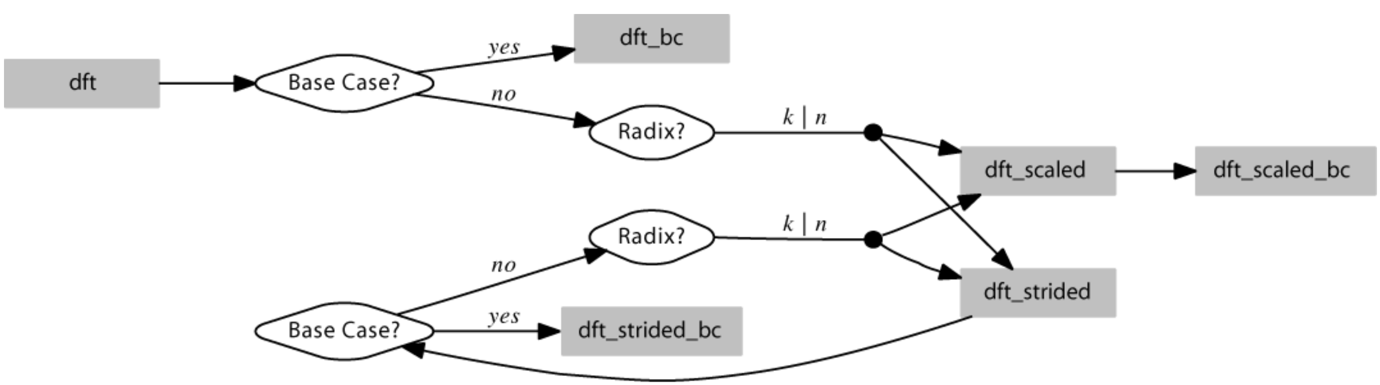

```
void dft(int n, cpx *y, cpx *x) {
    if (use_dft_base_case(n))
        dft_bc(n, y, x);
    else {
        int k = choose_dft_radix(n);
        for (int i=0; i < k; ++i)
            dft_strided(m, k, t + m*i, x + m*i);
        for (int i=0; i < m; ++i)
            dft_scaled(k, m, precomp_d[i],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

Decision Graph of Library

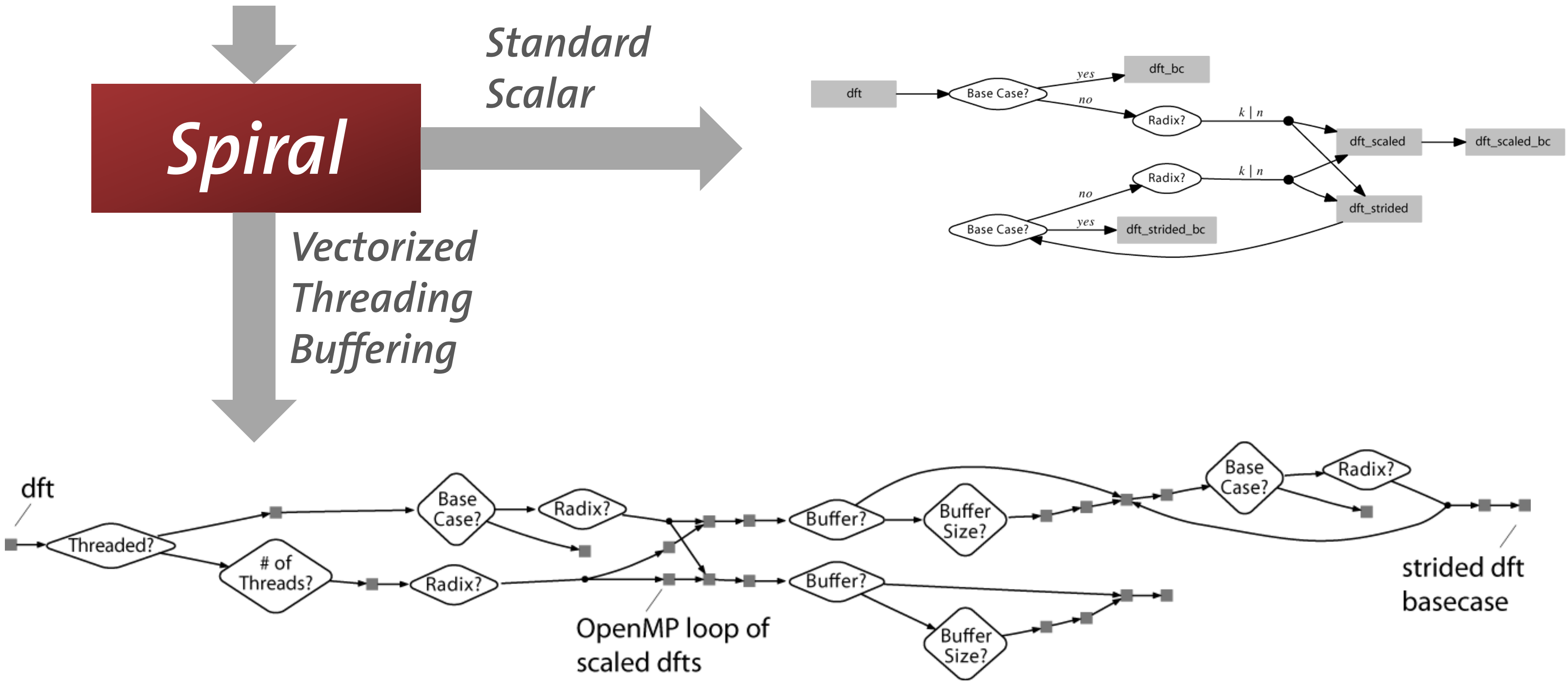
```
void dft(int n, cpx *y, cpx *x) {
    if (use_dft_base_case(n))
        dft_bc(n, y, x);
    else {
        int k = choose_dft_radix(n);
        for (int i=0; i < k; ++i)
            dft_strided(m, k, t + m*i, x + m*i);
        for (int i=0; i < m; ++i)
            dft_scaled(k, m, precomp_d[i],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



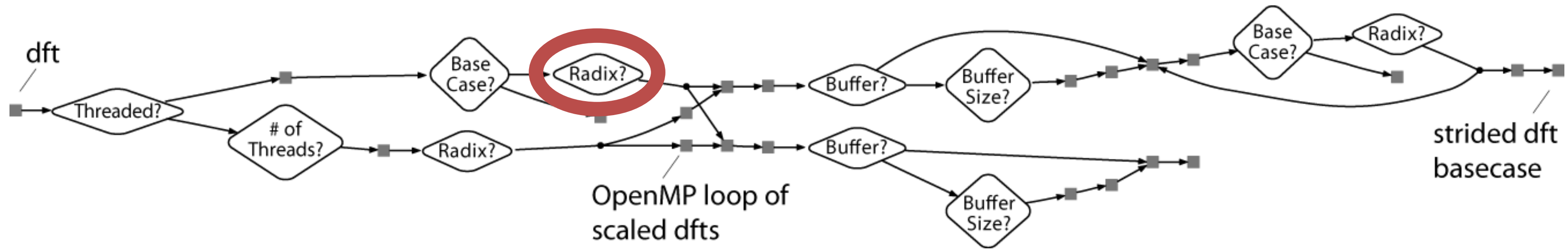
Spiral-Generated Libraries

$$(\mathbf{DFT}_k \otimes \mathrm{I}_m)\, \mathsf{T}^n_m (\mathrm{I}_k \otimes \mathbf{DFT}_m)\, \mathsf{L}^n_k$$



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

Our Work



Upon installation, generate decision trees for each choice

Example:

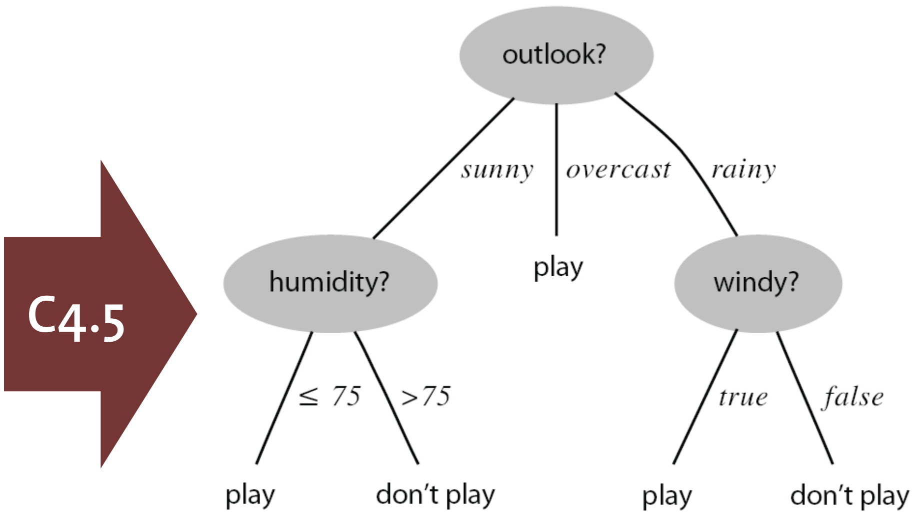
```
if ( n <= 65536 ) {
   if ( n <= 32 ) {
      if ( n <= 4 ) {return 2;}
      else {return 4;}
   }
   else {
      if ( n <= 1024 ) {
         if ( n <= 256 ) {return 8;}
         else {return 32;}
      }
      else {

      ..................
```

Statistical Classification: C4.5

Features (events)

Outlook	Temperature	Humidity	Windy	Decision
sunny	85	85	false	don't play
sunny	80	90	true	don't play
overcast	83	78	false	play
rain	70	96	false	play
rain	68	80	false	play
rain	65	70	true	don't play
overcast	64	65	true	play
sunny	72	95	false	don't play
sunny	69	70	false	play
rain	75	80	false	play
sunny	75	70	true	play
overcast	72	90	true	play
overcast	81	75	false	play
rain	71	80	true	don't play



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)**

```
dft(int n, cpx *y, cpx *x)
dft_strided(int n, int istr, cpx *y, cpx *x)
dft_scaled(int n, int str, cpx *d, cpx *y, cpx *x)
```

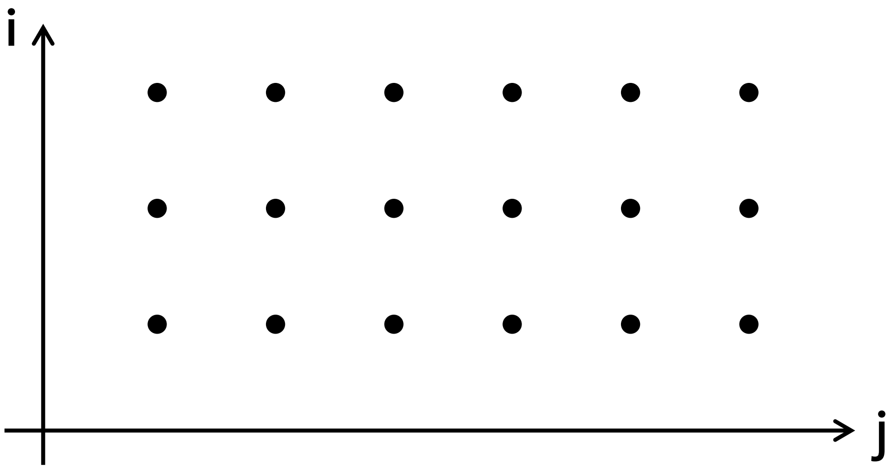
- **At installation time:**
  - Run search for a few input sizes n
  - Yields training set: features and associated decisions (several for each size)
  - Generate decision trees using C4.5 and insert into library

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**

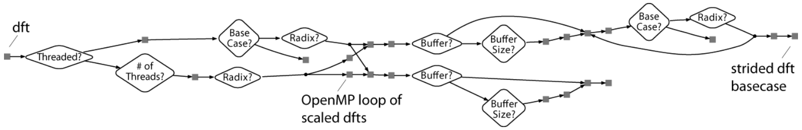
$$n = 2^i 3^j = 2, 3, 4, 6, 9, 12, 16, 18, 24, 27, 32, \ldots$$

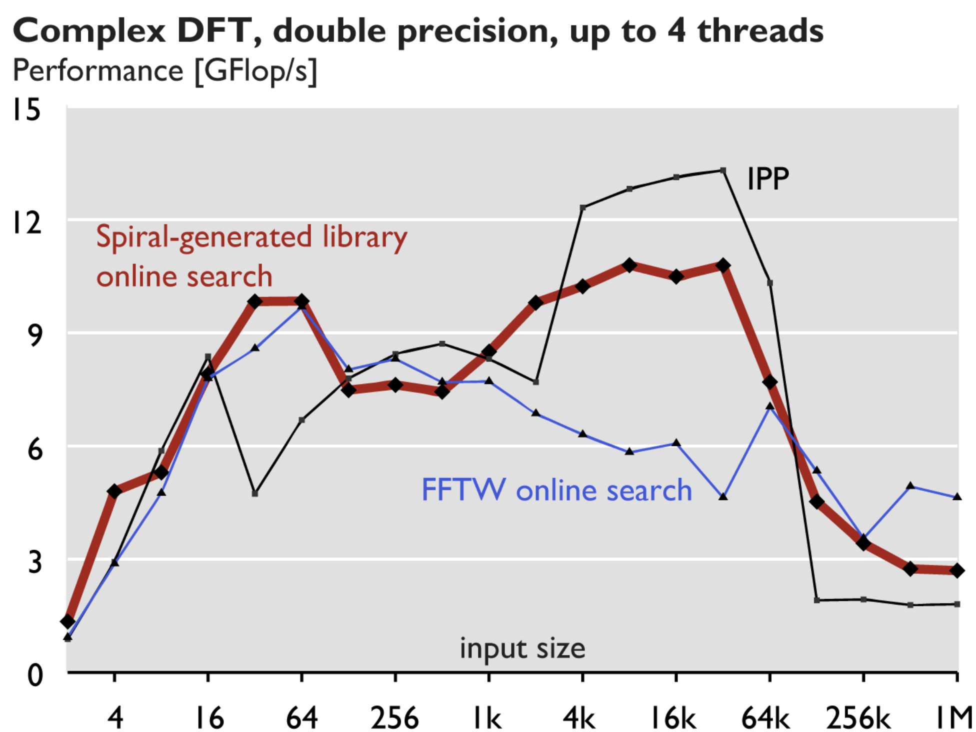


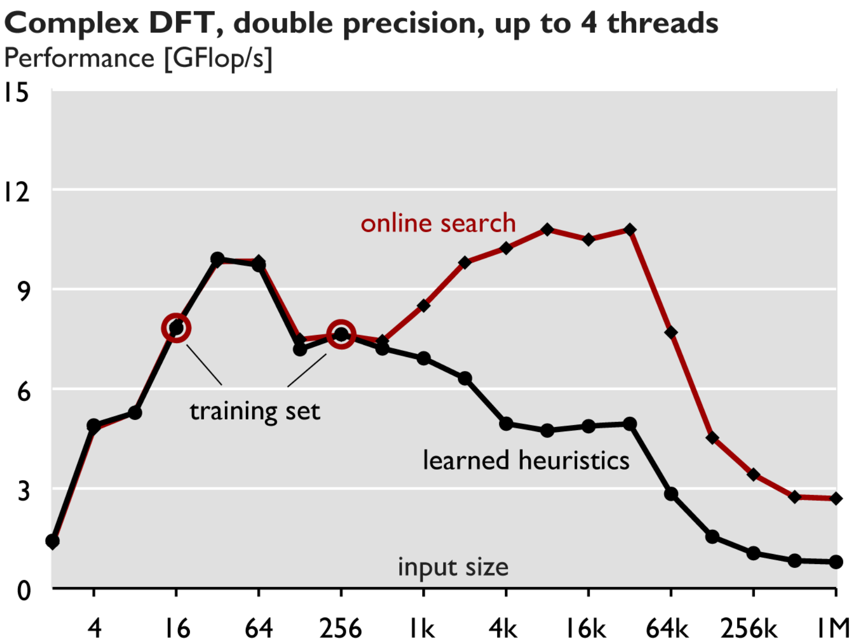
Compute i, j
and add to features

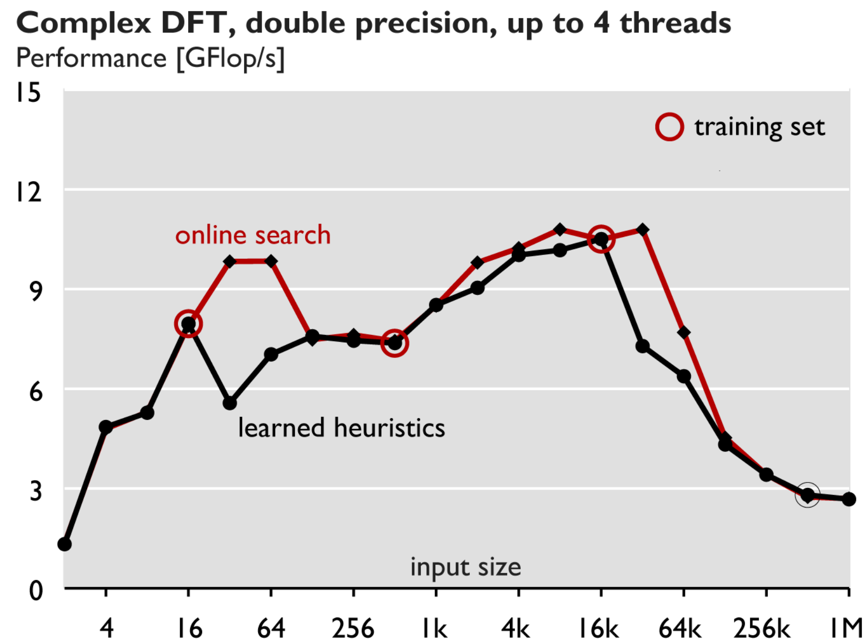
Experimental Setup

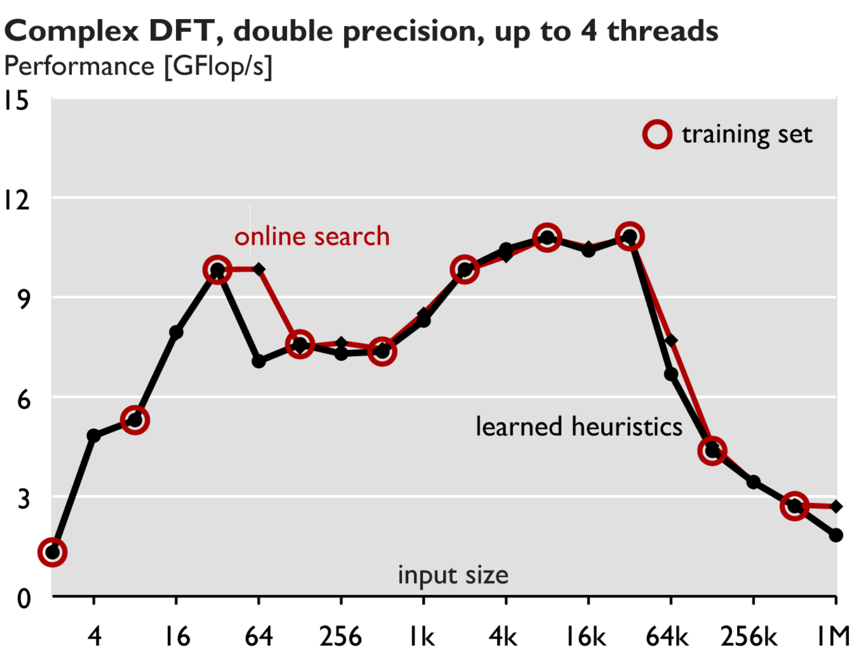
- 3GHz Intel Xeon 5160 (2 Core 2 Duos = 4 cores)
- Linux 64-bit, icc 10.1
- Libraries:
  - IPP 5.3
  - FFTW 3.2 alpha 2
  - Spiral-generated library





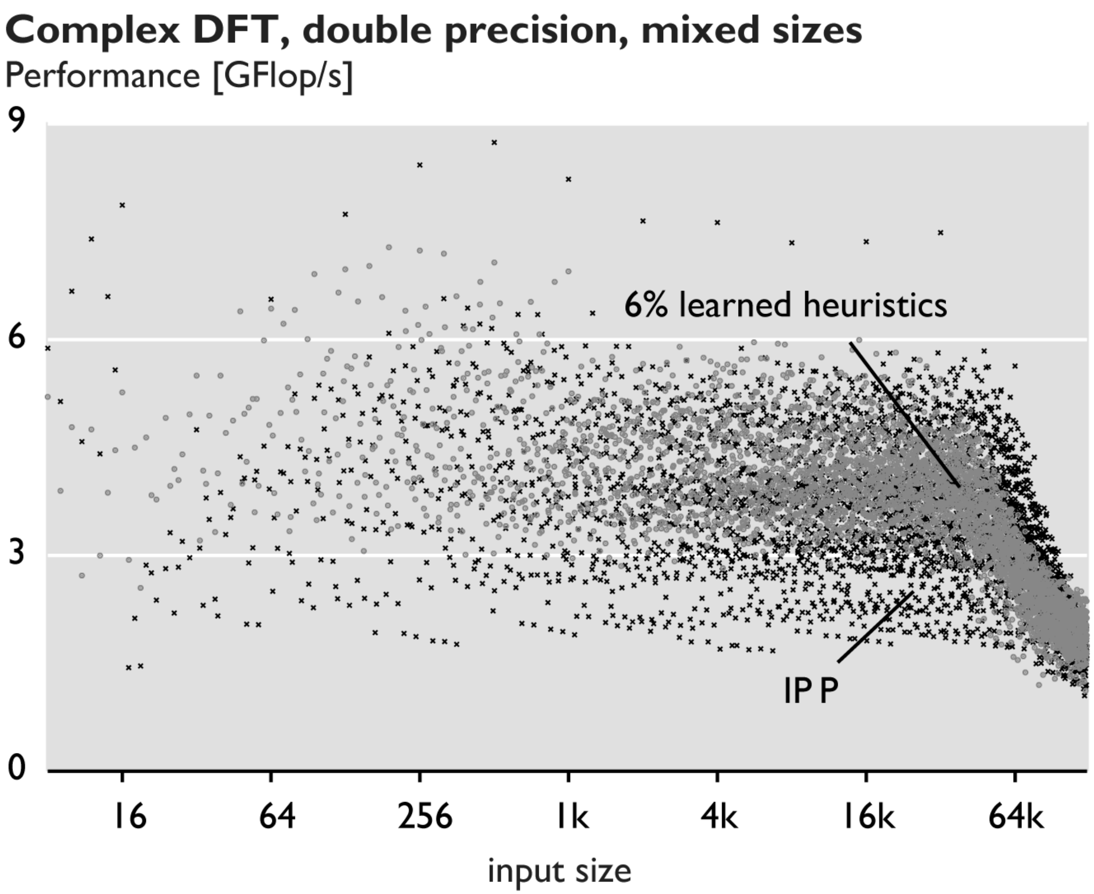






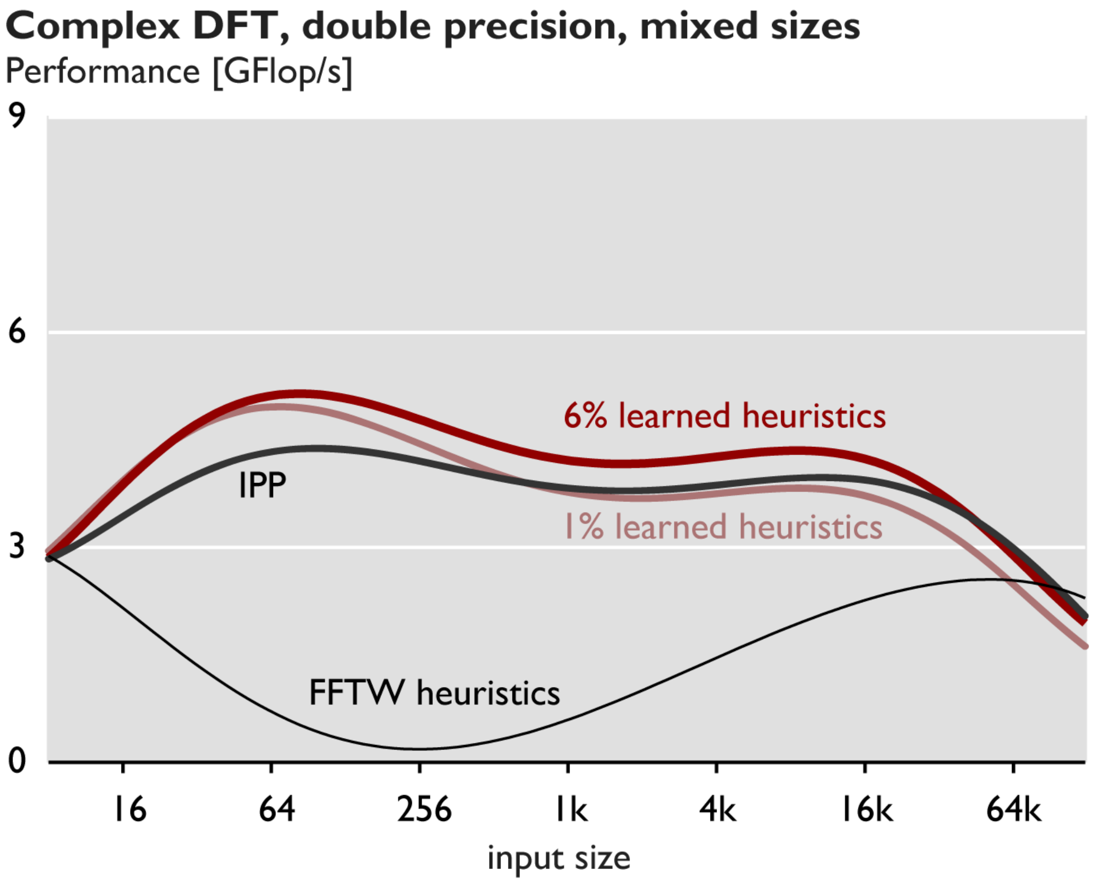
Learning works as expected

“All” Sizes



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

“All” Sizes

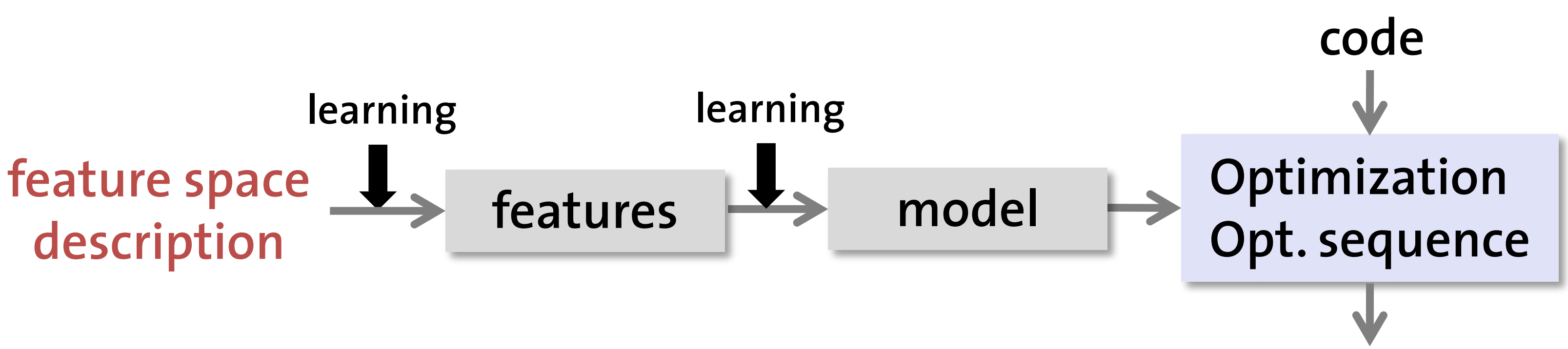


- All sizes $n \leq 2^{18}$, with prime factors ≤ 19
- Higher order fit of all sizes

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
  Proc. International Conference on Machine Learning (ICML), pp. 729-736, 2009

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