

Automatic Performance Tuning and Machine Learning

Markus Püschel
Computer Science, ETH Zürich

with:

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

ETH

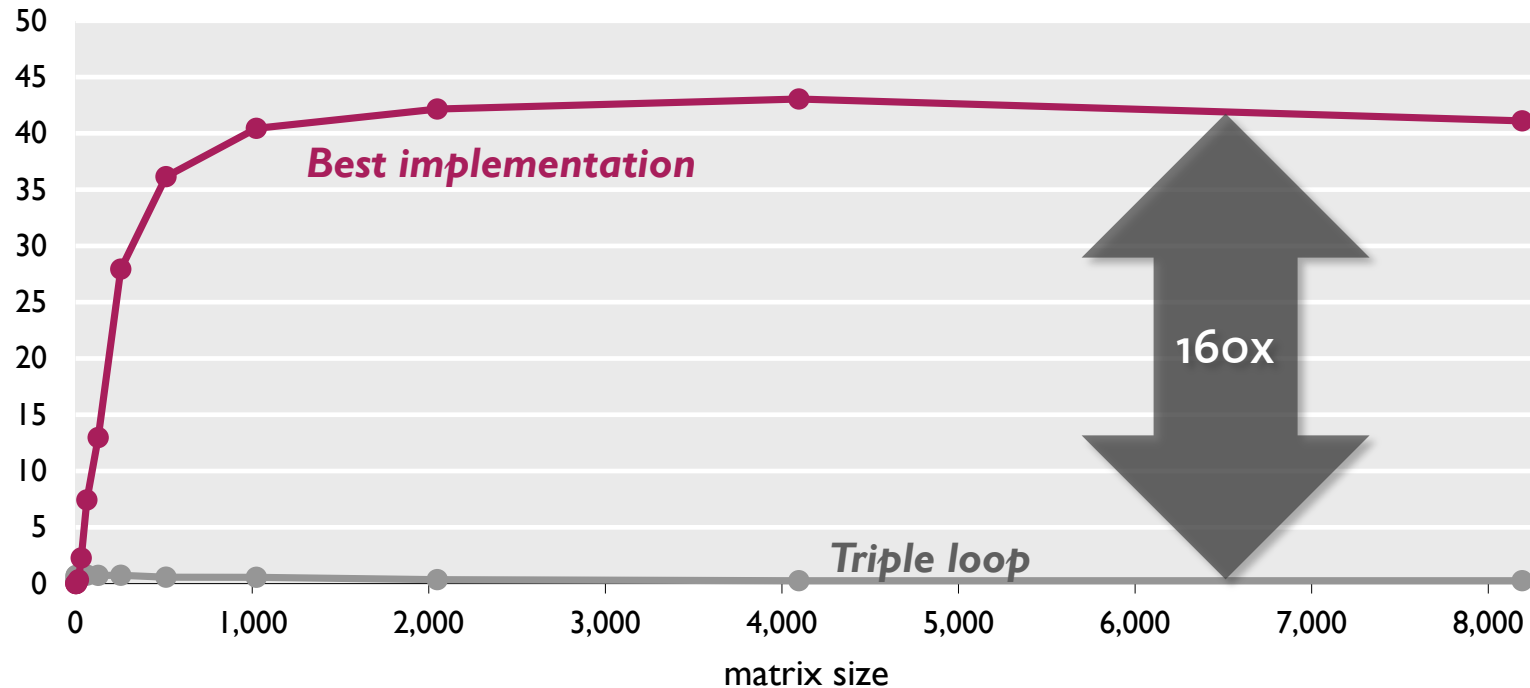
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SPIRAL 
www.spiral.net

Why Autotuning?

Matrix-Matrix Multiplication (MMM) on quadcore Intel platform

Performance [Gflop/s]

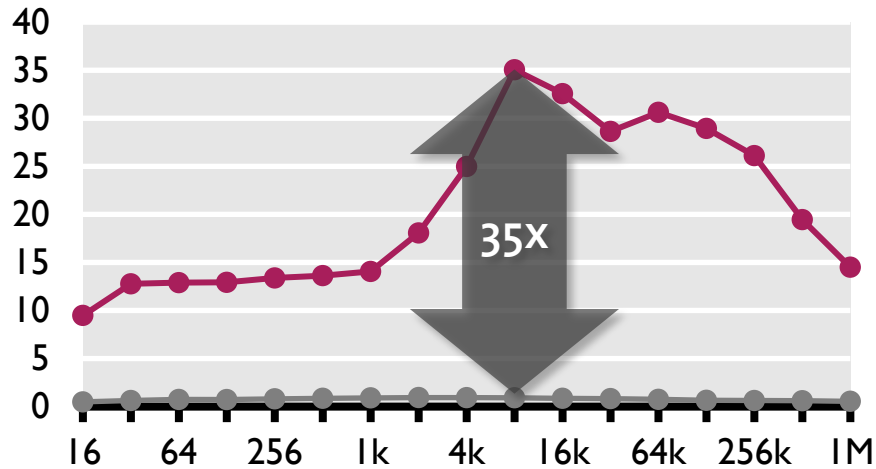


- Same (mathematical) operation count ($2n^3$)
- Compiler underperforms by 160x

Same for All Critical Compute Functions

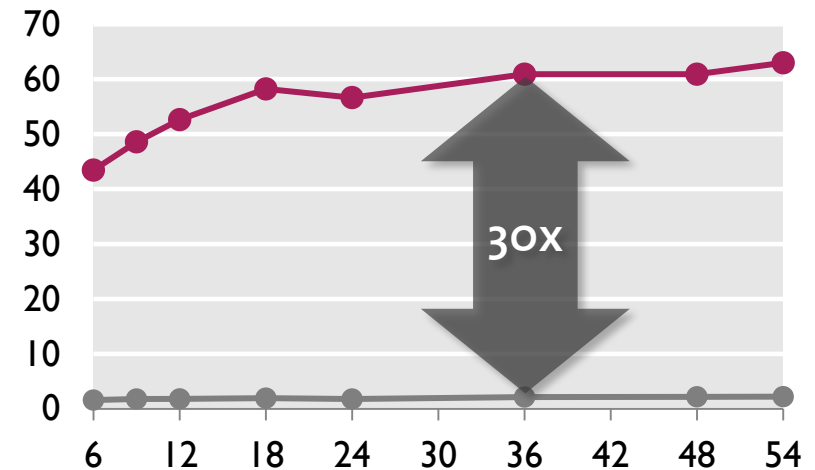
DFT auf Intel Core i7

Leistung [Gflop/s]



WiFi Empfänger (1 Intel Core)

Leistung [Mbit/s]



Solution: Autotuning

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

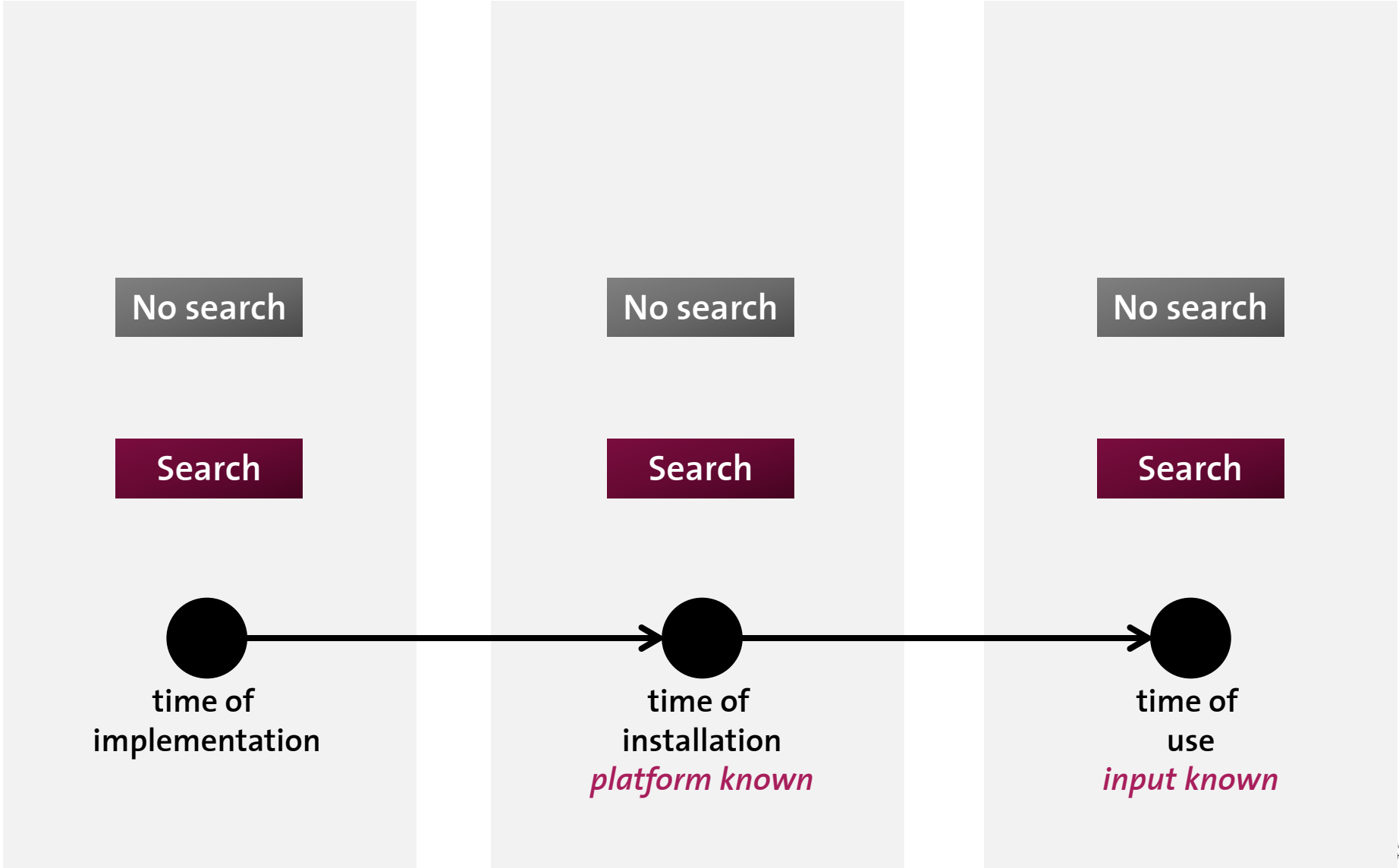
Definition: *Automating performance optimization with tools that complement/aid the compiler or programmer*

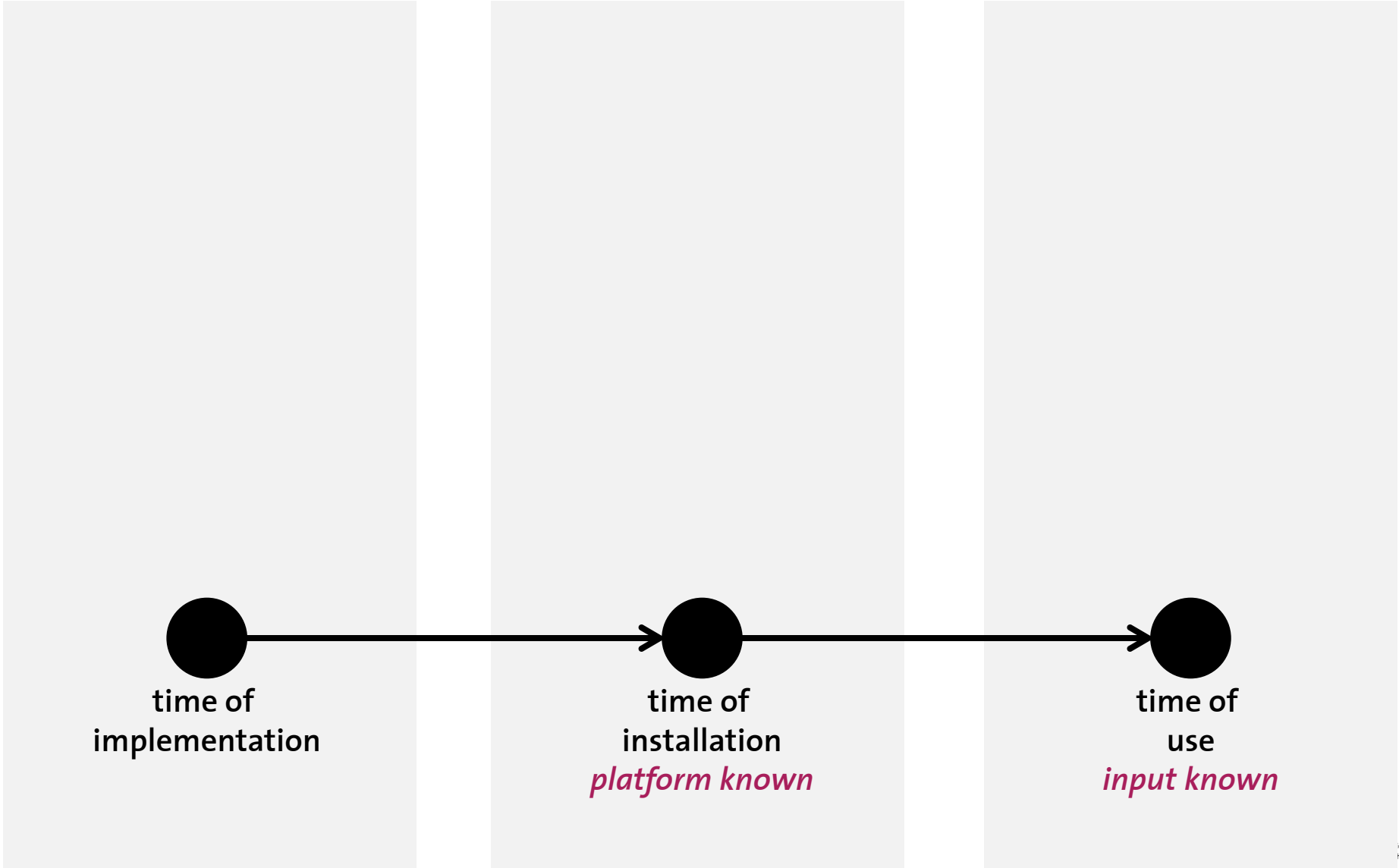
However: *Search is an important tool. But expensive.*

Solution: Machine learning

Organization

- Autotuning examples
- An example use of machine learning





PhiPac/ATLAS: MMM Generator

Whaley, Bilmes, Demmel, Dongarra, ...

```
// MMM loop-nest
```

```
for i = 0:NB:N-1
```

```
  for j = 0:NB:M-1
```

```
    for k = 0:NB:K-1
```

- *ijk or jik depending on N and M*
- *Blocking for cache*

```
// mini-MMM loop nest
```

```
for i' = i:MU:i+NB-1
```

```
  for j' = j:NU:j+NB-1
```

```
    for k' = k:KU:k+NB-1
```

- *Blocking for registers*

```
// micro-MMM loop nest
```

```
for k'' = k':1:k'+KU-1
```

```
  for i'' = i':1:i'+MU-1
```

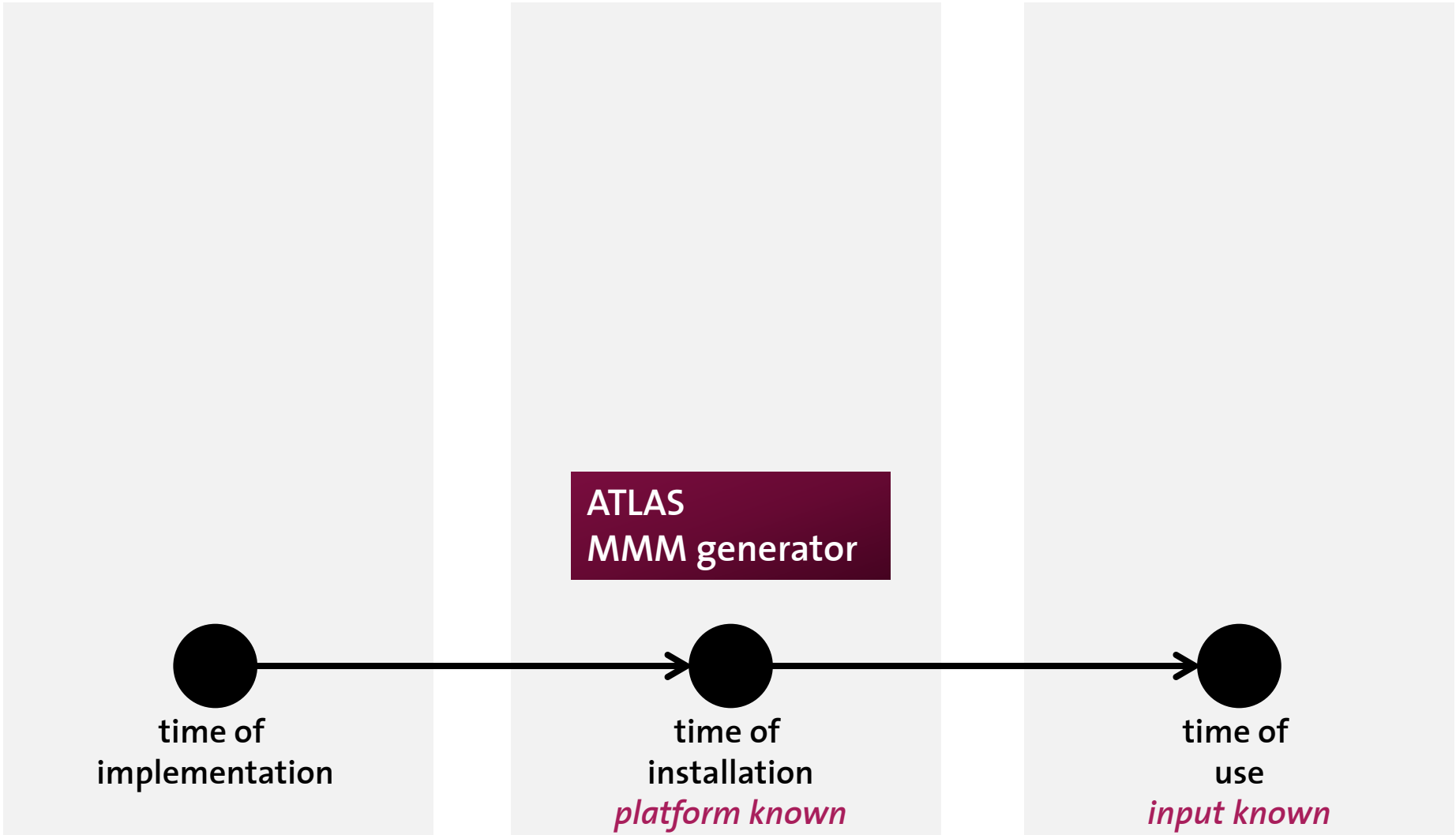
```
    for j'' = j':1:j'+NU-1
```

- *Unrolling*
- *Scalar replacement*
- *Add/mult interleaving*
- *Skewing*

Search parameters: N_B, M_U, N_U, K_U, L_S, ...

No search

Search



FFTW: Discrete Fourier Transform (DFT)

Frigo, Johnson

Installation

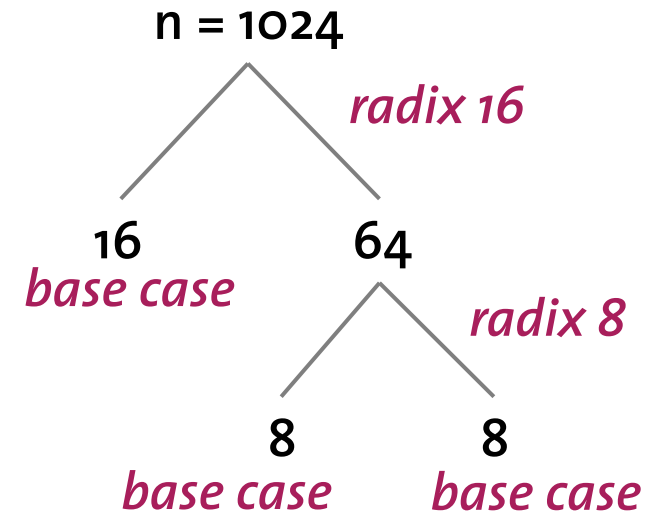
configure/make

Usage

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

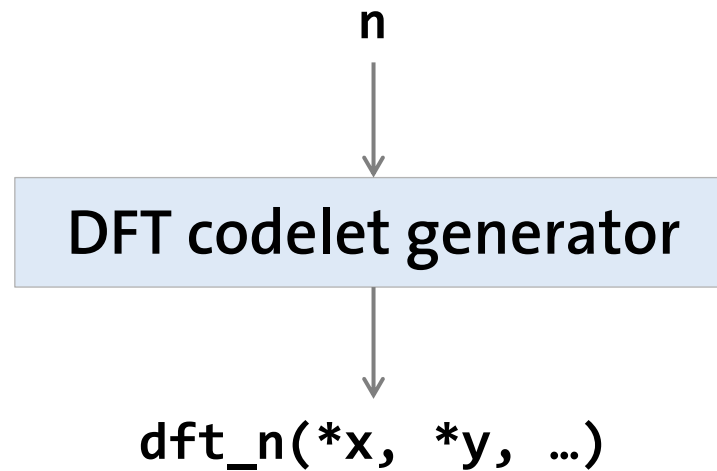
Twiddles

Search for fastest
computation strategy



FFTW: Codelet Generator

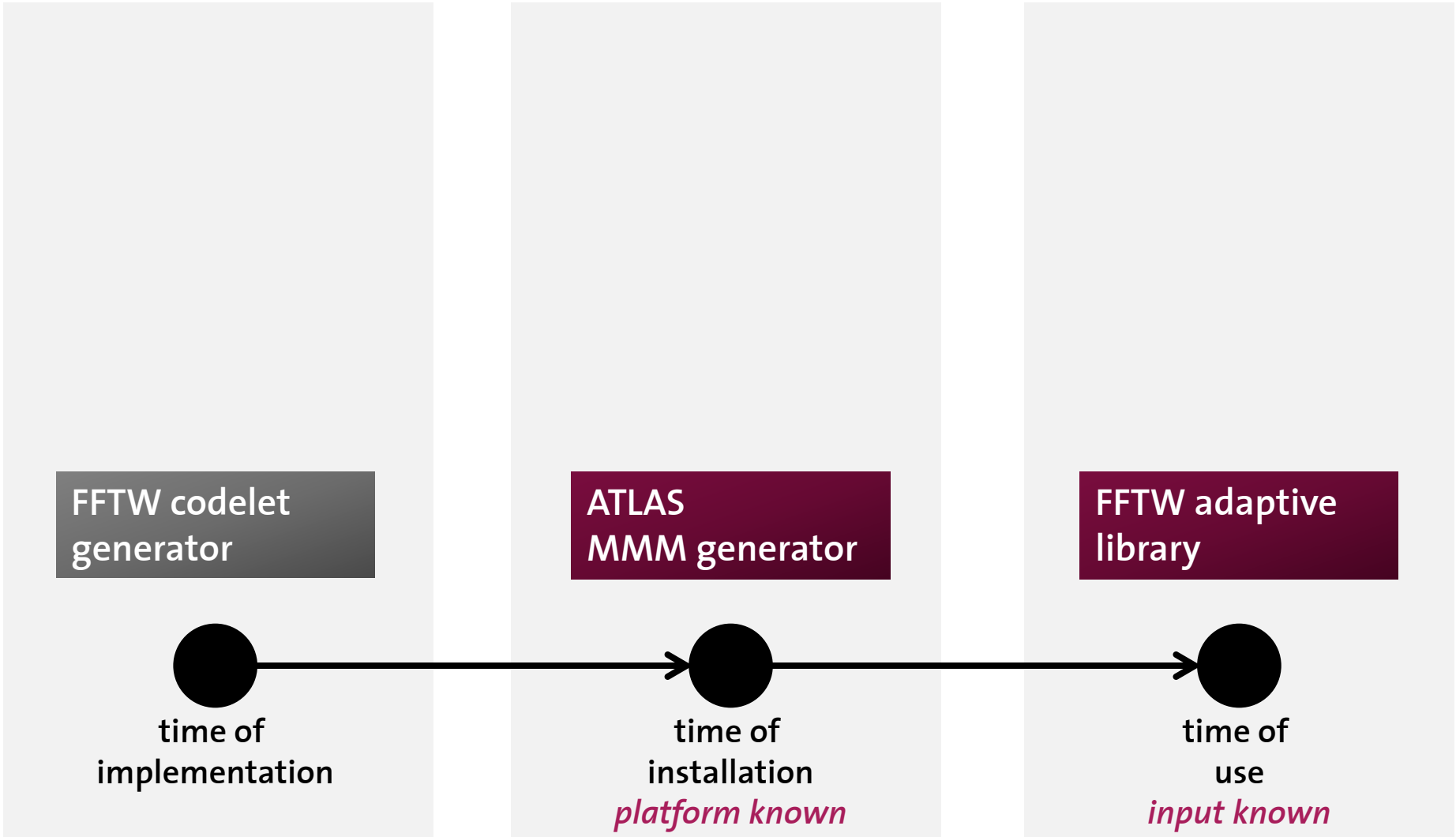
Frigo



*fixed size DFT function
straightline code*

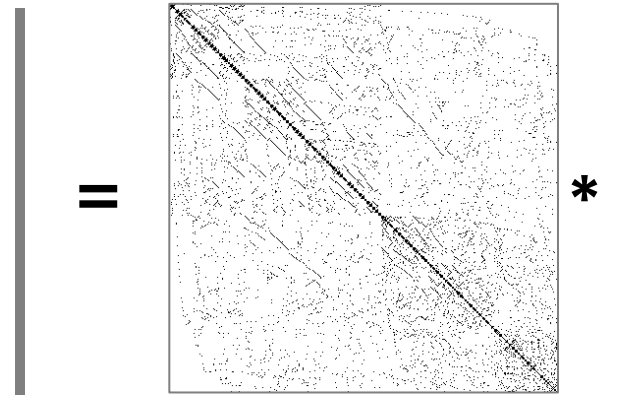
No search

Search



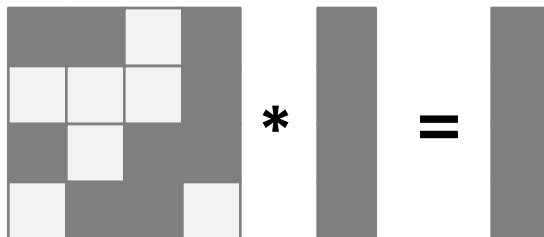
OSKI: Sparse Matrix-Vector Multiplication

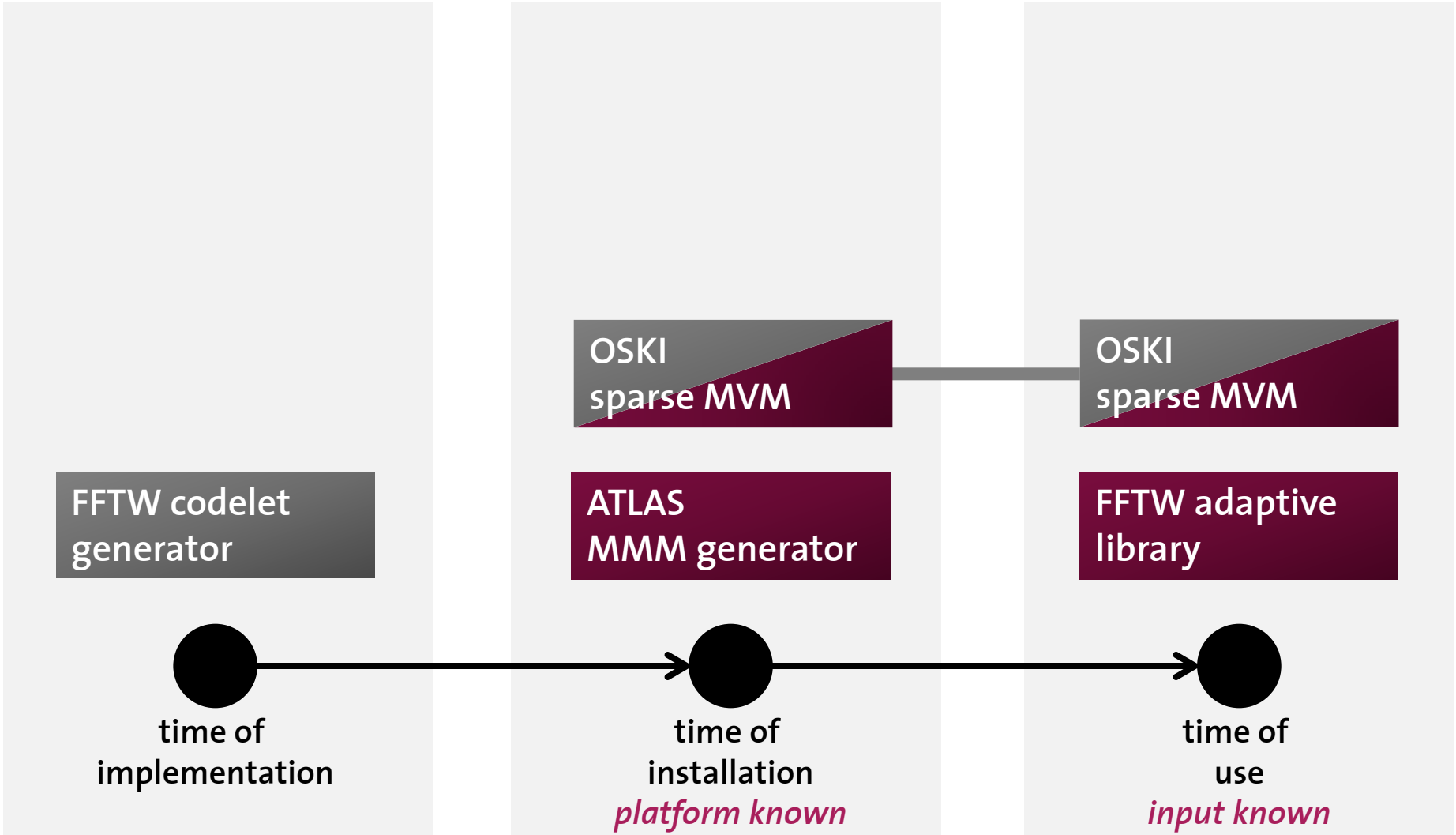
Vuduc, Im, Yelick, Demmel



■ Blocking for registers:

- Improves locality (reuse of input vector)
- But creates overhead (zeros in block)





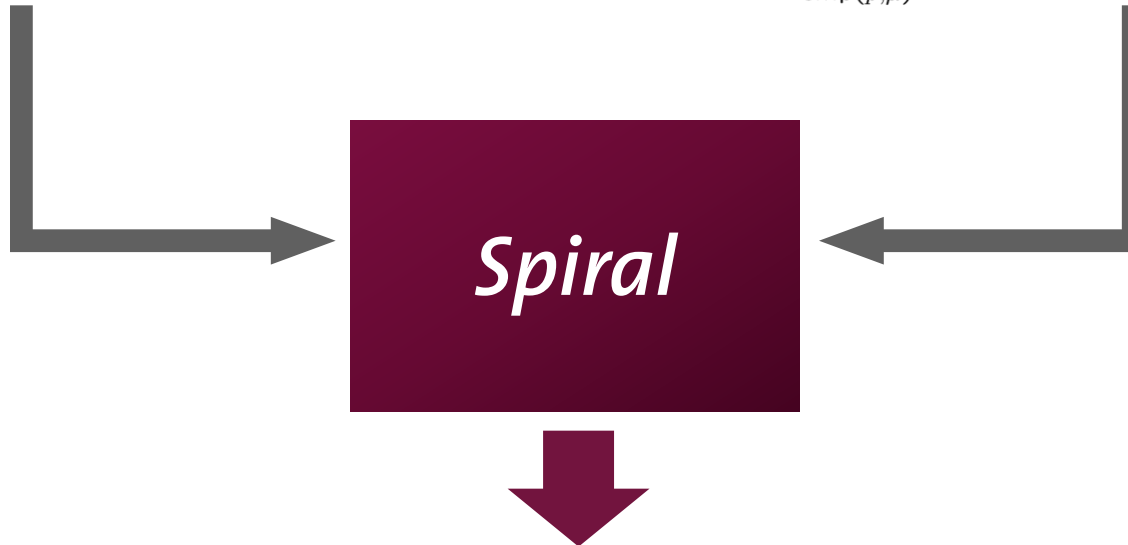
Spiral: Linear Transforms & More

Algorithm knowledge

$$\begin{aligned} \text{DFT}_n &\rightarrow P_{k/2,2m}^\top \left(\text{DFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i C_{2m} \text{rDFT}_{2m}(i/k) \right) \right) \left(\text{RDFT}'_k \otimes I_m \right) \\ \begin{vmatrix} \text{rDFT}_{2n}(u) \\ \text{rDHT}_{2n}(u) \end{vmatrix} &\rightarrow L_m^{2n} \left(I_k \otimes_i \begin{vmatrix} \text{rDFT}_{2m}((i+u)/k) \\ \text{rDHT}_{2m}((i+u)/k) \end{vmatrix} \right) \left(\begin{vmatrix} \text{rDFT}_{2k}(u) \\ \text{rDHT}_{2k}(u) \end{vmatrix} \otimes I_m \right) \\ \text{RDFT-3}_n &\rightarrow (Q_{k/2,m}^\top \otimes I_2) (I_k \otimes_i \text{rDFT}_{2m}(i+1/2/k)) (\text{RDFT-3}_k \otimes I_m) \end{aligned}$$

Platform description

$$\begin{aligned} \underbrace{A_m \otimes I_n}_{\text{smp}(p,\mu)} &\rightarrow \underbrace{\left(L_m^{mp} \otimes I_{n/p} \right) \left(I_p \otimes (A_m \otimes I_{n/p}) \right) \left(L_p^{mp} \otimes I_{n/p} \right)}_{\text{smp}(p,\mu)} \\ \underbrace{I_m \otimes A_n}_{\text{smp}(p,\mu)} &\rightarrow I_p \otimes_{\parallel} \left(I_{m/p} \otimes A_n \right) \\ \underbrace{(P \otimes I_n)}_{\text{smp}(p,\mu)} &\rightarrow \left(P \otimes I_{n/\mu} \right) \overline{\otimes} I_\mu \end{aligned}$$



Optimized implementation
regenerated for every new platform

Program Generation in Spiral (Sketched)

Transform
user specified

DFT₈

↓ *Algorithm rules*

$$(DFT_2 \otimes I_4) T_4^8 (I_2 \otimes ((DFT_2 \otimes I_2) \cdot T_2^4 (I_2 \otimes DFT_2) L_2^4)) L_2^8$$



Fast algorithm
in SPL
many choices

Σ-SPL

$$\sum (S_j DFT_2 G_j) \sum \left(\sum (S_{k,l} \text{diag}(t_{k,l}) DFT_2 G_l) \sum (S_m \text{diag}(t_m) DFT_2 G_{k,m}) \right)$$



Optimized implementation

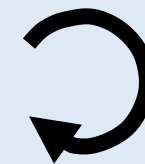
Optimization at all abstraction levels



parallelization
vectorization



loop
optimizations



constant folding
scheduling
.....

+ search

No search
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
input known

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

$(\text{DFT}_k \otimes I_m) \mathbb{T}_m^n (I_k \otimes \text{DFT}_m) \mathbb{L}_k^n$
+ some platform information

Spiral

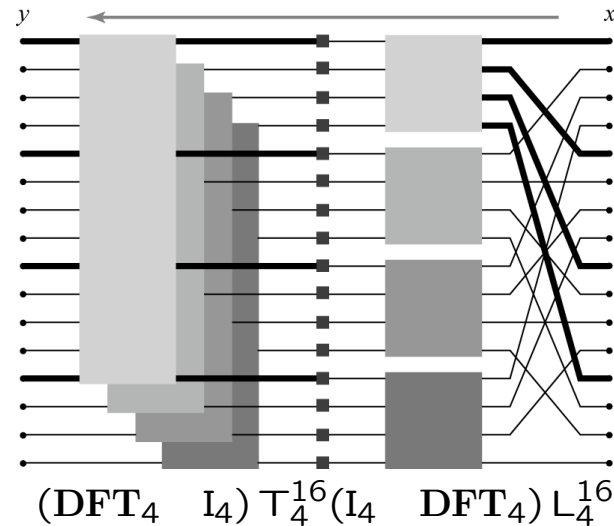
Online tunable library

$$\begin{aligned}
 \text{DFT}_n &\rightarrow P_{k/2,2m}^\top (\text{DFT}_{2m} \oplus (I_{k/2-1} \ i \ C_{2m} \text{rDFT}_{2m}(i/k))) (\text{RDFT}'_k \ I_m), \quad k \text{ even,} \\
 \begin{bmatrix} \text{RDFT}'_n \\ \text{RDFT}'_n \\ \text{DHT}'_n \\ \text{DHT}'_n \end{bmatrix} &\rightarrow (P_{k/2,2m}^\top \ I_2) \left(\begin{bmatrix} \text{RDFT}'_{2m} \\ \text{RDFT}'_{2m} \\ \text{DHT}'_{2m} \\ \text{DHT}'_{2m} \end{bmatrix} \oplus \left(I_{k/2-1} \ i \ D_{2m} \begin{bmatrix} \text{rDFT}'_{2m}(i/k) \\ \text{rDFT}'_{2m}(i/k) \\ \text{rDHT}'_{2m}(i/k) \\ \text{rDHT}'_{2m}(i/k) \end{bmatrix} \right) \right) \left(\begin{bmatrix} \text{RDFT}'_k \\ \text{RDFT}'_k \\ \text{DHT}'_k \\ \text{DHT}'_k \end{bmatrix} \ I_m \right), \quad k \text{ even,} \\
 \begin{bmatrix} \text{rDFT}_{2n}(u) \\ \text{rDHT}_{2n}(u) \end{bmatrix} &\rightarrow L_m^{2n} \left(I_k \ i \ \begin{bmatrix} \text{rDFT}_{2m}((i+u)/k) \\ \text{rDHT}_{2m}((i+u)/k) \end{bmatrix} \right) \left(\begin{bmatrix} \text{rDFT}_{2k}(u) \\ \text{rDHT}_{2k}(u) \end{bmatrix} \ I_m \right), \\
 \text{RDFT-3}_n &\rightarrow (Q_{k/2,m}^\top \ I_2) (I_k \ i \ \text{rDFT}_{2m}(i+1/2)/k) (\text{RDFT-3}_k \ I_m), \quad k \text{ even,} \\
 \text{DCT-2}_n &\rightarrow P_{k/2,2m}^\top (\text{DCT-2}_{2m} K_2^{2m} \oplus (I_{k/2-1} \ N_{2m} \text{RDFT-3}_{2m}^\top)) B_n(L_{k/2}^{n/2} \ I_2) (I_m \ \text{RDFT}'_k) Q_{m/2,k}, \\
 \text{DCT-3}_n &\rightarrow \text{DCT-2}_n^\top, \\
 \text{DCT-4}_n &\rightarrow Q_{k/2,2m}^\top (I_{k/2} \ N_{2m} \text{RDFT-3}_{2m}^\top) B'_n(L_{k/2}^{n/2} \ I_2) (I_m \ \text{RDFT-3}_k) Q_{m/2,k}. \\
 \text{DFT}_n &\rightarrow (\text{DFT}_k \ I_m) \mathbb{T}_m^n (I_k \ \text{DFT}_m) \mathbb{L}_k^n, \quad n = km \\
 \text{DFT}_n &\rightarrow P_n (\text{DFT}_k \ \text{DFT}_m) Q_n, \quad n = km, \text{ gcd}(k, m) = 1 \\
 \text{DFT}_p &\rightarrow R_p^\top (I_1 \oplus \text{DFT}_{p-1}) D_p (I_1 \oplus \text{DFT}_{p-1}) R_p, \quad p \text{ prime} \\
 \text{DCT-3}_n &\rightarrow (I_m \oplus J_m) \mathbb{L}_m^n (\text{DCT-3}_m(1/4) \oplus \text{DCT-3}_m(3/4)) \\
 &\quad \cdot (F_2 \ I_m) \begin{bmatrix} I_m & 0 \oplus -J_{m-1} \\ \frac{1}{\sqrt{2}}(I_1 \oplus 2I_m) \end{bmatrix}, \quad n = 2m \\
 \text{DCT-4}_n &\rightarrow S_n \text{DCT-2}_n \text{diag}_{0 \leq k < n} (1/(2 \cos((2k+1)\pi/4n))) \\
 \text{IMDCT}_{2m} &\rightarrow (J_m \oplus I_m \oplus I_m \oplus J_m) \left(\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \ I_m \right) \oplus \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} \ I_m \right) \right) J_{2m} \text{DCT-4}_{2m} \\
 \text{WHT}_{2^k} &\rightarrow \prod_{i=1}^t (I_2^{k_1+\dots+k_{i-1}} \ \text{WHT}_{2^{k_i}} \ I_2^{k_{i+1}+\dots+k_t}), \quad k = k_1 + \dots + 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{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*

Fast Fourier Transform



$$16 = 4 \times 4$$

```

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);
    }
}

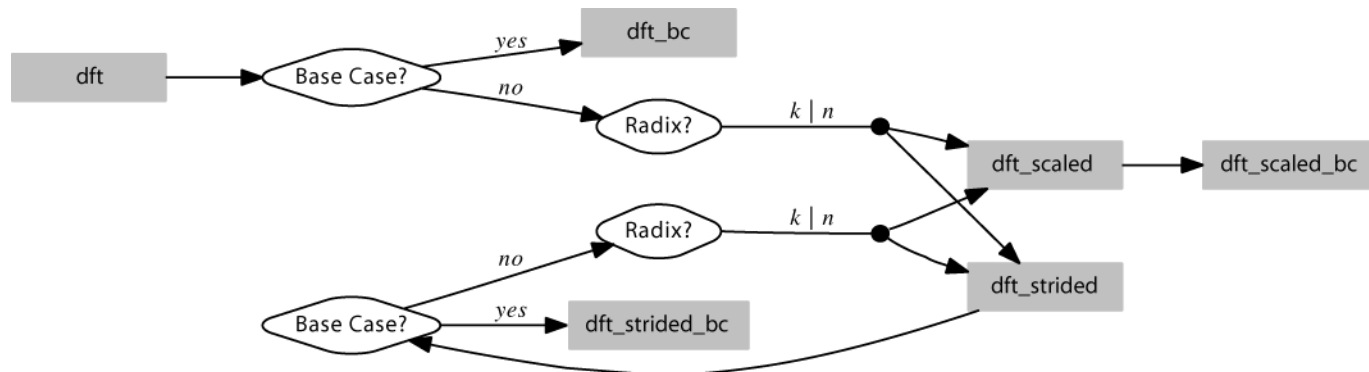
void dft_strided(int n, int istr, cpx *y, cpx *x) { ... }
void dft_scaled(int n, int str, cpx *d, cpx *y, cpx *x) { ... }
    
```

Choices used for adaptation

Decision Graph of Library

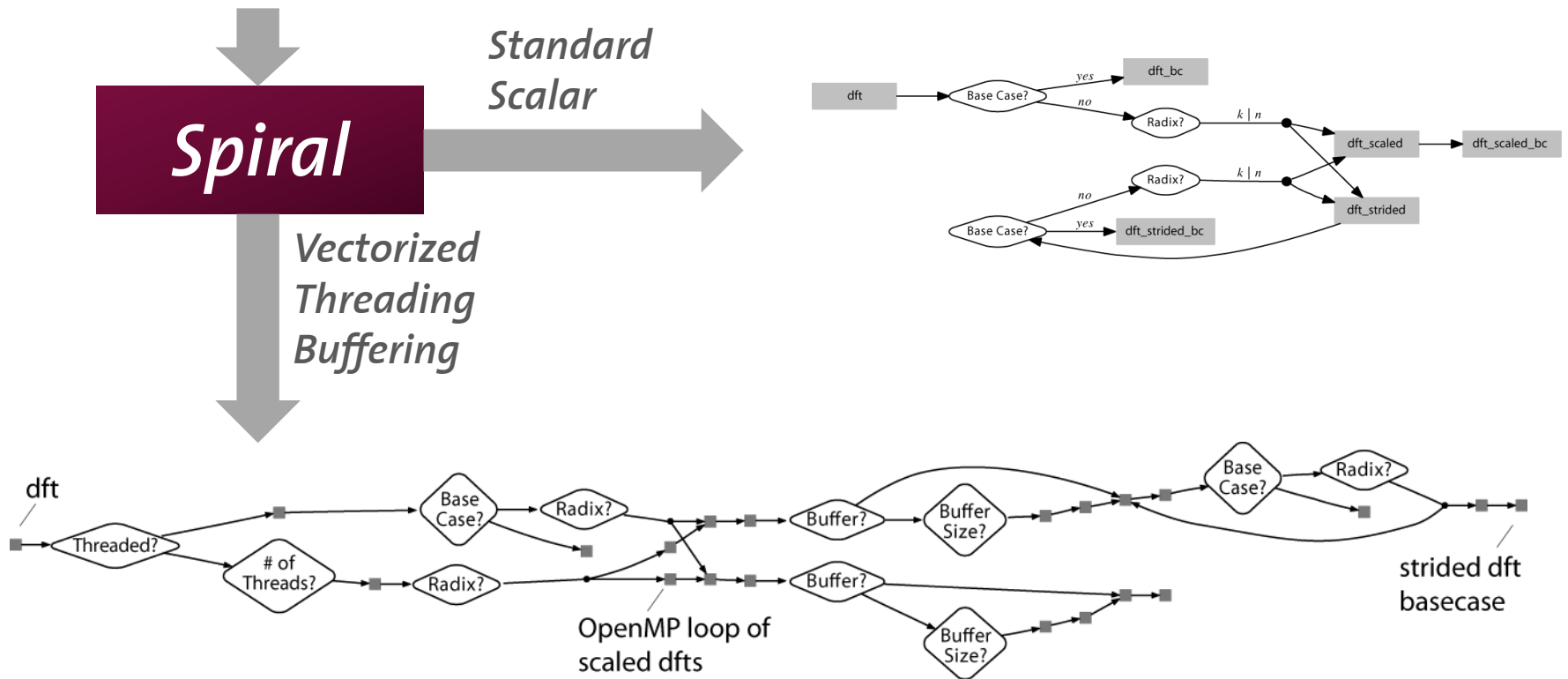
```
void dft(int n, cpx *y, cpx *x) {  
    if (use_dft_base_case(n))  
        dft_bc(n, y, x);  
    else {  
        int k = choose_dft_radix(n);  
        for (int i=0; i < k; ++i)  
            dft_strided(m, k, t + m*i, x + m*i);  
        for (int i=0; i < m; ++i)  
            dft_scaled(k, m,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



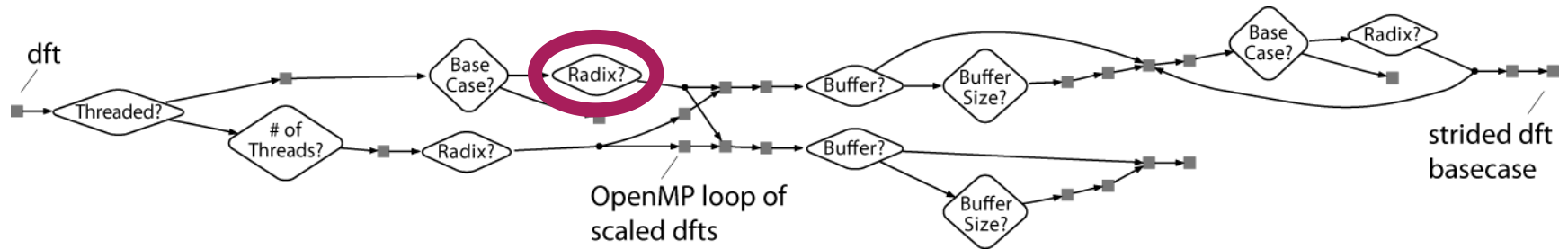
Spiral-Generated Libraries

$$(\text{DFT}_k \otimes \text{I}_m) \text{T}_m^n (\text{I}_k \otimes \text{DFT}_m) \text{L}_k^n$$



- 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:

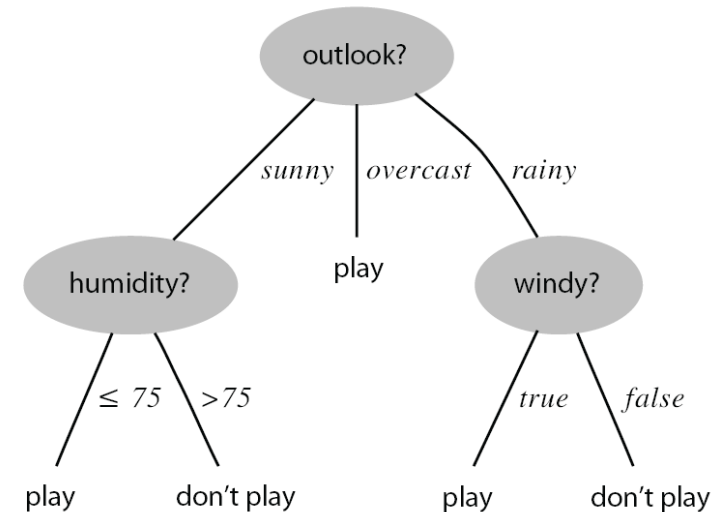
```
if ( n <= 65536 ) {  
  if ( n <= 32 ) {  
    if ( n <= 4 ) {return 2;}  
    else {return 4;}  
  }  
  else {  
    if ( n <= 1024 ) {  
      if ( n <= 256 ) {return 8;}  
      else {return 32;}  
    }  
    else {
```

.....

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



$$P(\text{play}|\text{windy}=\text{false}) = 6/8$$

$$P(\text{don't play}|\text{windy}=\text{false}) = 2/8$$

$$P(\text{play}|\text{windy}=\text{true}) = 1/2$$

$$P(\text{don't play}|\text{windy}=\text{true}) = 1/2$$



Entropy of Features

$$H(\text{windy}) = 0.89$$

$$\mathbf{H(\text{outlook}) = 0.69}$$

$$H(\text{humidity}) = \dots$$

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)
```

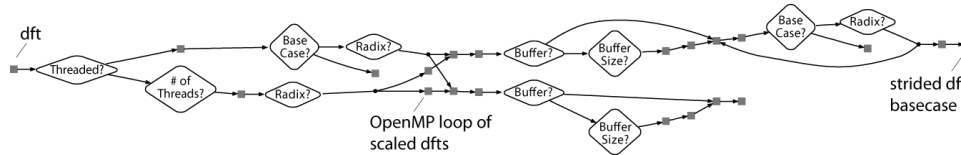
```
dft_scaled(int n, int str, cpx *d, cpx *y, cpx *x)
```

- 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
- Some issues resolved (de Mesmay et al. 2010)

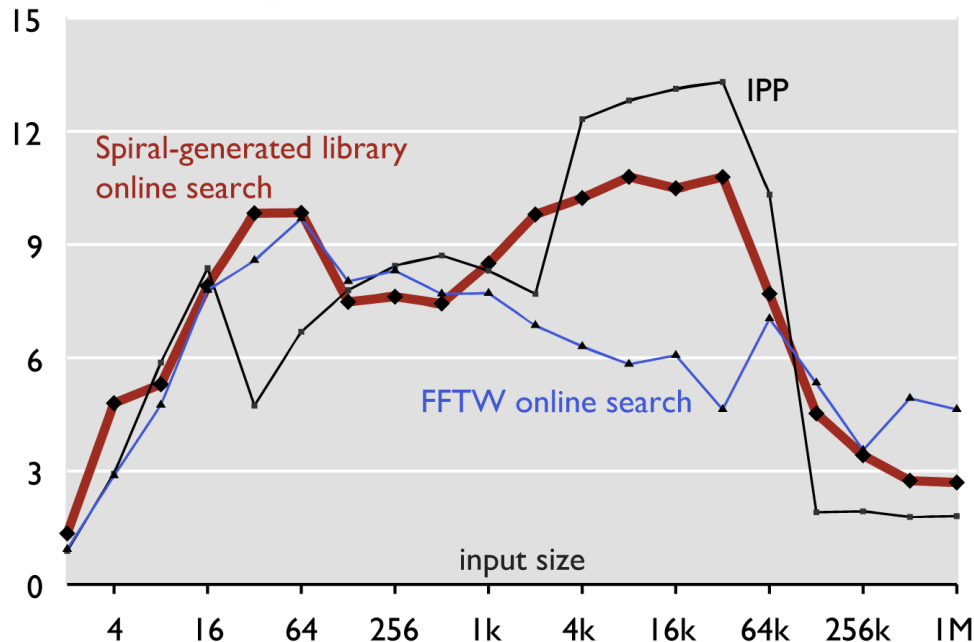
Experimental Setup

- 3GHz Intel Xeon 5160 (2 Core 2 Duos = 4 cores), icc 10.1
- Spiral-generated library:



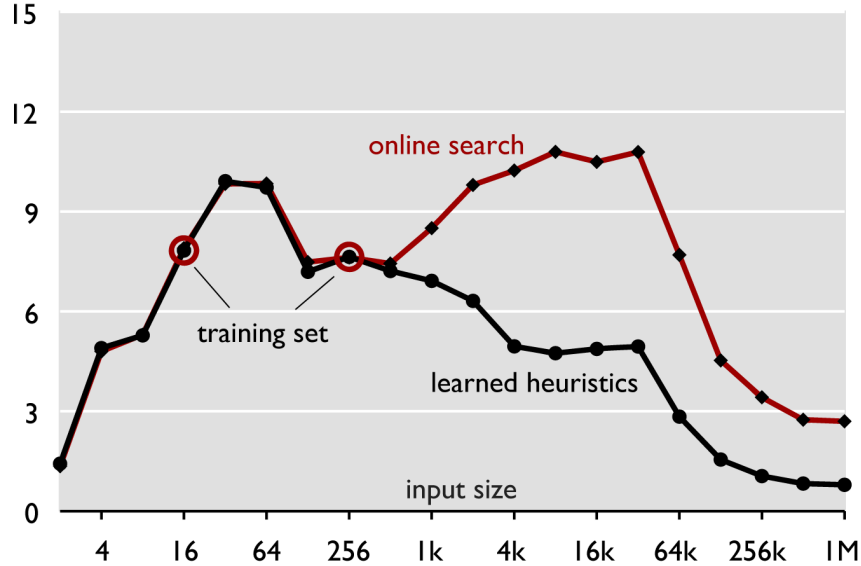
Complex DFT, double precision, up to 4 threads

Performance [GFlop/s]



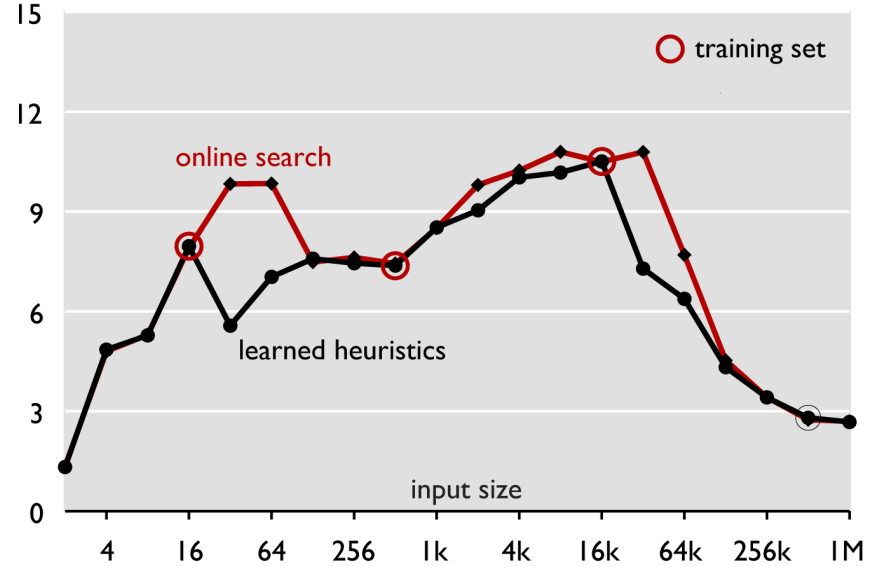
Complex DFT, double precision, up to 4 threads

Performance [GFlop/s]



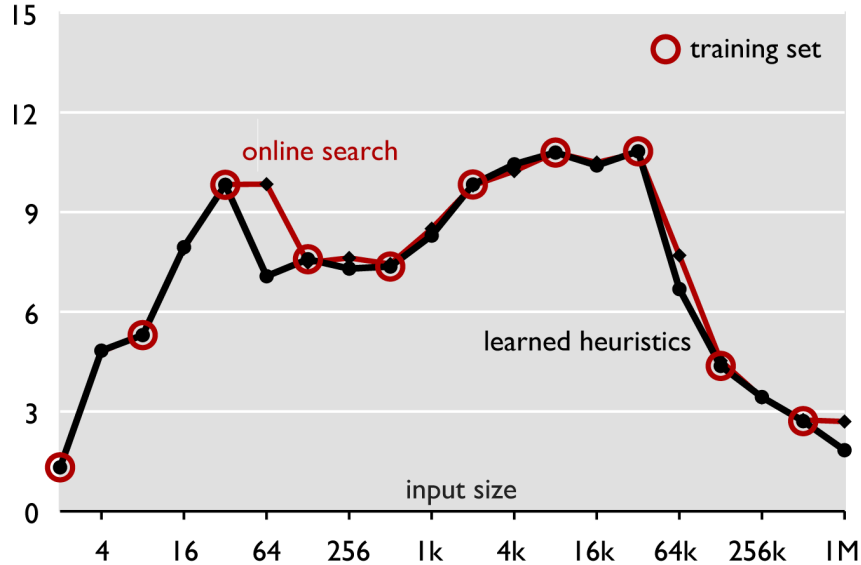
Complex DFT, double precision, up to 4 threads

Performance [GFlop/s]



Complex DFT, double precision, up to 4 threads

Performance [GFlop/s]

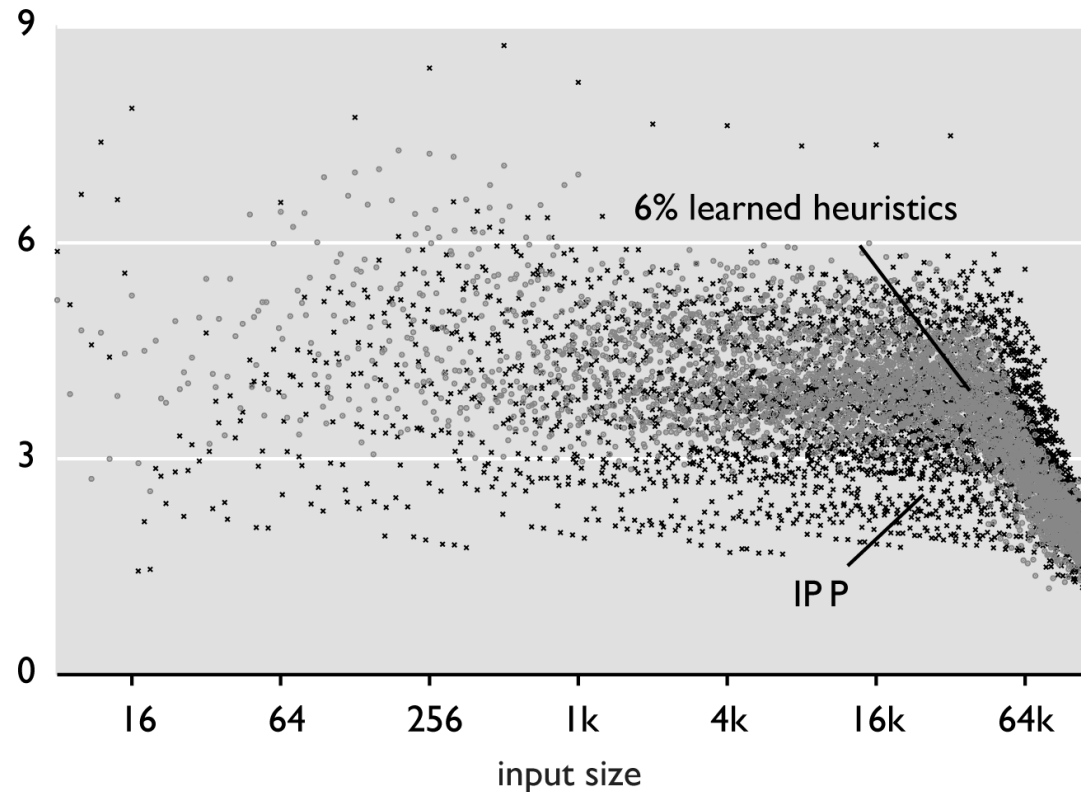


Learning works as expected

“All” Sizes

Complex DFT, double precision, mixed sizes

Performance [GFlop/s]

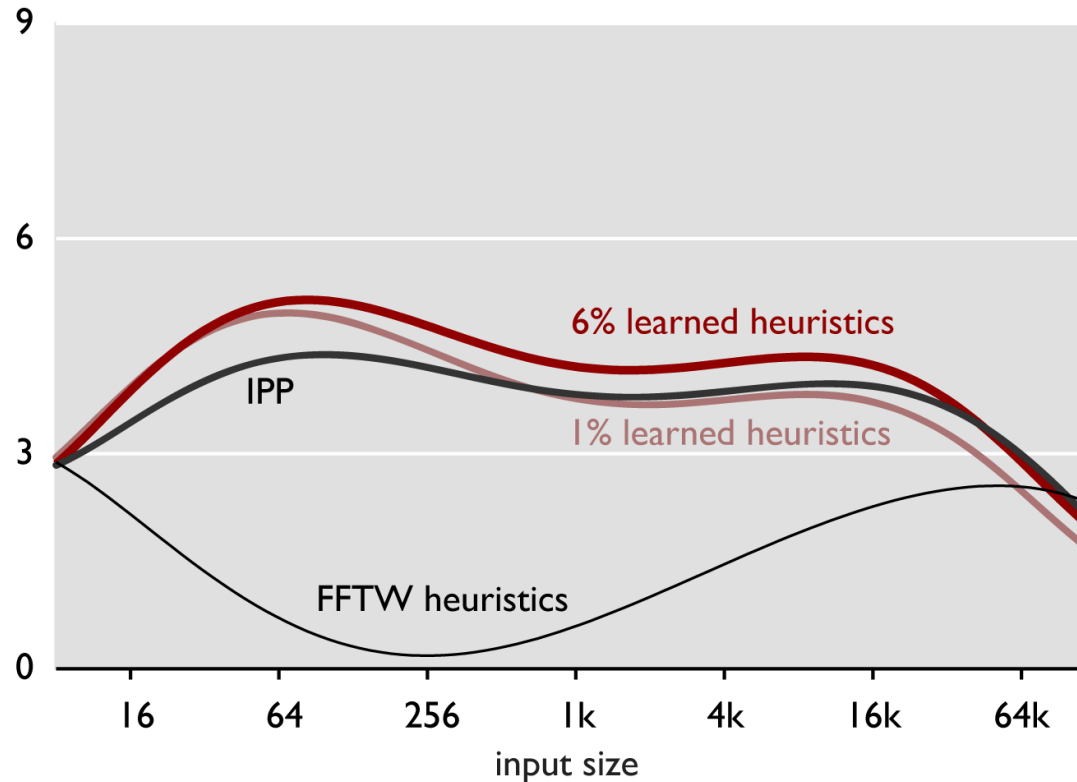


- All sizes $n \leq 2^{18}$, with prime factors ≤ 19

“All” Sizes

Complex DFT, double precision, mixed sizes

Performance [GFlop/s]



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

Related Work

■ Machine learning in autotuning

- Linear regression for stencil/sorting (Brewer 1995)
- *Pioneering work:*
 - Learning DFT recursions in Spiral (Singer/Veloso 2000)
- Linear regression/SVM in PhiPAC (Vuduc/Demmel/Bilmes 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)

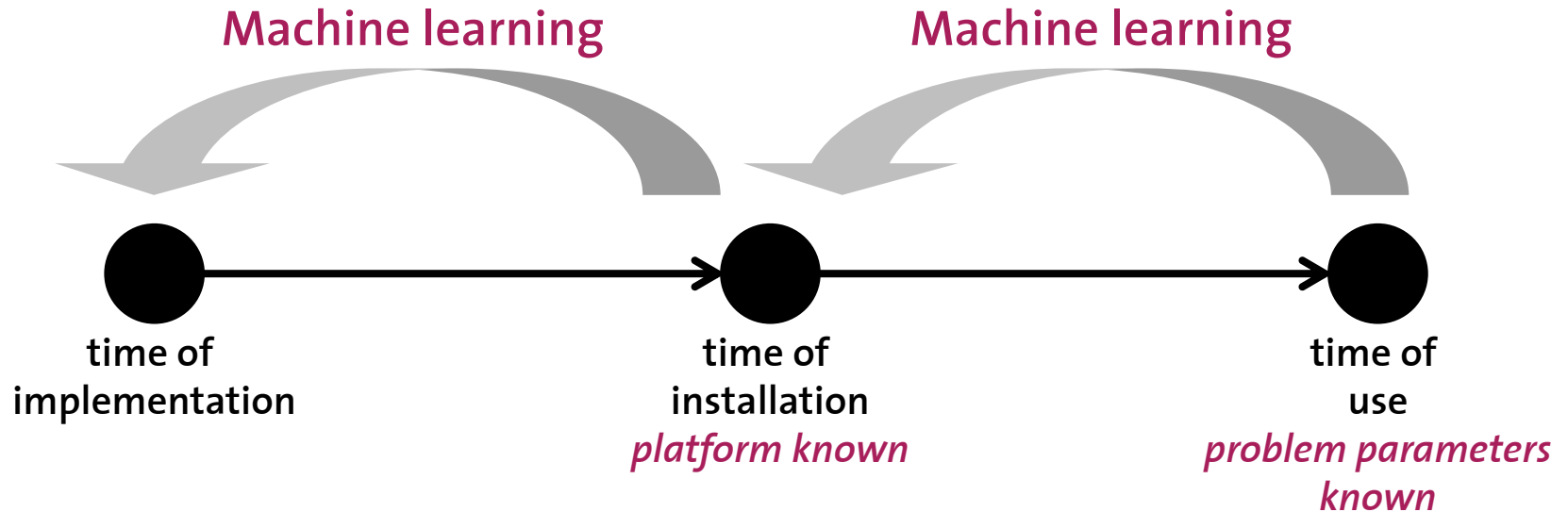
This Talk

- 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

Other Recent ML work in Spiral

- 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
- **Don't forget: autotuning \neq search**