# 第１０回先進スーパーコンピューティング環境研究会（ASE 研究会）実施報告

東京大学情報基盤センター　特任准教授　片桐孝洋

2011 年 10 月 13 日（木）15 時 00 分～18 時 30 分、東京大学情報基盤センター４階遠隔講義室にて、第 10 回先進スーパーコンピューティング環境研究会（ASE 研究会）が開催されました。

国内の大学･研究機関および企業からの参加者が 13 名あり､活発な議論がなされました。

招待講演として、ETH Zurich から Markus Püschel 教授をお呼びし、自動チューニングと機械学習に関する講演を行いました。また、工学院大学の藤井昭宏 博士、筑波大学の山崎育朗様による、数値計算アルゴリズムにおける自動チューニングの講演を行いました。

Püschel 教授の講演は、「Automatic Performance Tuning and Machine Learning」と題し、自動チューニングの重要性などの背景に始まり、自動チューニングへ機械学習を適用した事例紹介に至る大変興味深いご講演でした。特に、自動チューニングを行うことは「探索」処理に位置づけられることが多いのですが、Püschel 教授の講演では、自動チューニングはコンパイラやプログラマを助けるためのツールが本来の意味であり探索はコスト高である、ですから機械学習を適用する解を示すという主張は明快であり、わかりやすいものでした。Püschel 教授らが行ってきた信号処理（FFT）の自動チューニングに関するプロジェクト Spiral における自動チューニングの位置づけも大変興味深いものでした。参加者間で、たいへん活発な議論がなされました。

以上の講演内容について、本原稿の後に当日の発表資料を掲載します。ご興味のある方はご覧ください。当日のプログラムを以下に載せます。

---

## Program

- 15:00-15:05　Introduction　by　Takahiro Katagiri (The University of Tokyo)
- 15:05 – 16:05

**Invited Speaker:** Professor Markus Püschel (Computer Science, ETH Zurich, Switzerland)

**Title:** Automatic Performance Tuning and Machine Learning

**Abstract:** Automatic performance tuning has emerged as a paradigm complementing traditional compilers to port software and performance between platforms. Several techniques have proven useful including adaptive libraries, program generation, domain-specific languages, and models. However, one technique is shared by almost all approaches: search for the fastest among a set of alternative implementations. Typically the search space is huge and hence the search is costly. This may be bearable in offline tuning (e.g., ATLAS) that is performed during installation but becomes cumbersome in online tuning (e.g., FFTW) that is performed at runtime since the input

size is required. We argue that machine learning, which has already been studied and used in the compiler community, can solve this problem and should be added to the portfolio of performance tuning tools. As example we show a successful approach to automatically convert Spiral-generated online-tunable transform libraries into offline-tunable ones.

- 16:05-16:15 Break
- 16:15 – 16:45

**Sparker:** Akihiro Fujii (Kogakuin University), Osamu Nakamura(Sumitomo Metal Industries)

**Title:** Automatic tuning for Algebraic Multigrid solver for Fluid analysis

**Abstract:** This talk presents an online automatic tuning method of Algebraic Multigrid (AMG) solver for fluid analysis based on SMAC method. This type of fluid analysis requires to solve Pressure Poisson equation every time step. As time steps forward, problem matrix does not change, but the right hand side vector changes gradually in many cases. To shorten the total time the solver uses, we try to optimize AMG solver by selecting its parameter setting for each time step. In this talk, we consider automatic tuning technique of AMG solver parameters such as smoothers, multigrid cycles, acceleration coefficient of the smoother and others. We studied the auto tuning method which narrows the search domain by determining parameters in a step-by-step manner based on typical property of the AMG solver parameters. In addition, optimizing parameters is done every constant number of time steps such as 100 time steps to reduce the overhead of measuring the efficiency of various parameter settings. Our AMG library with online auto-tuning mechanism which determines appropriate solver parameter set among 900 parameter settings improved the performance of AMG solver with default setting up to 20 percent in our numerical tests.

- 16:45 – 17:15

**Speaker:** Ikuro Yamazaki, Hiroto Tadano, Tetsuya Sakurai (Graduate School of Systems and Information Engineering, University of Tsukuba)

**Title:** A parameter selection for a preconditioner using a cutoff for Krylov subspace methods

**Abstract:** Large linear systems involving a semi-sparse matrix which has relatively large number of nonzero elements appear in nano-science simulations. Preconditioners with a cutoff is proposed as suitable methods for such semi-sparse linear systems. The performance of these preconditioners is highly dependent on a cutoff parameter. A smaller value of cutoff increases a computational cost in preconditioner construction. Meanwhile, a larger value of cutoff leads to a less effective

preconditioning matrix with a large number of iterations. Hence, the selection of an appropriate cutoff parameter is important. In this talk, we present a strategy to find an efficient cutoff parameter for the preconditioners using a cutoff. We estimate an appropriate cutoff parameter using residual norms before applying an iterative solver, and verify the validity of our strategy by numerical experiments.

- 17:15 – 17:30 Break
- 17:30 -18: 00

**Speaker:** Satoshi Ohshima (The University of Tokyo)

**Title:** Implementation of 3D FEM program on GPU

**Abstract:** GPU is now utilizing for several scientific applications because GPU has high calculation performance and memory transfer performance. FEM (Finite Element Method) is one of the applications expected to accelerate with GPU. This talk shows the implementation of 3D FEM program on CUDA GPU. The main targets of acceleration are sparse matrix solver (CG method) and matrix assembly. Our GPU implementation has obtained better performance than multi-core CPU. Also we are now trying multi GPUs implementation.

- 18:00 - 18:30

**Speaker:** Masae Hayashi (The University of Tokyo)

**Title:** OpenMP/MPI Hybrid Parallel FEM Based on Extended Hierarchical Interface Decomposition for Multi-core Clusters.

**Abstract:** ILU preconditioner is a powerful and popular preconditioning method for Krylov iterative solvers on sparse matrices derived from FEM applications. Block-Jacobi-type localized ILU preconditioner is generally employed in parallel computation basing on domain decomposition. The localization of ILU process is good for parallel efficiency but decreases the effectiveness of preconditioning since it neglects the effect from out side the domain. Hierarchical Interface Decomposition (HID) and proposed extended version of HID, where additional layers of separators are introduced for robust and efficient computation is a robust and efficient parallel preconditioning method. We are developing preconditioning methods using OpenMP/MPI hybrid parallel programming models on multicore/multisocket clusters. HID and extended version of HID (ExHID) is applied to both of inter-node parallelization with message-passing (e.g. MPI) part and intra-node parallelization with multi-threading (e.g. OpenMP). We implement the OpenMP/MPI hybrid parallel programing model to FEM application solving 3- dimensional linear elasticity problem. The developed code has been tested on the T2K Open Super Computer (T2K/Tokyo) using up to 8 nodes, 16 cores. We report the scalability and the robustness resulted from numerical experiments of

parallel ILU preconditioner with fill-ins and iterative solver and we show the developed code provides better performance compared to that of multicoloring method used for intra-node parallelization.

- 18: 30 Closing Remarks by Takahiro Katagiri (The University of Tokyo)
- 19:00- Banquet near Nedu station

---


図１　当日の会場の様子

ASE 研究会の開催情報はメーリングリストで発信をしております。研究会メーリングリストに参加ご希望の方は、ASE 研究会幹事の片桐（katagiri@cc.u-tokyo.ac.jp）までお知らせください。

以上

# Automatic Performance Tuning and Machine Learning

Markus Püschel
Computer Science, ETH Zürich

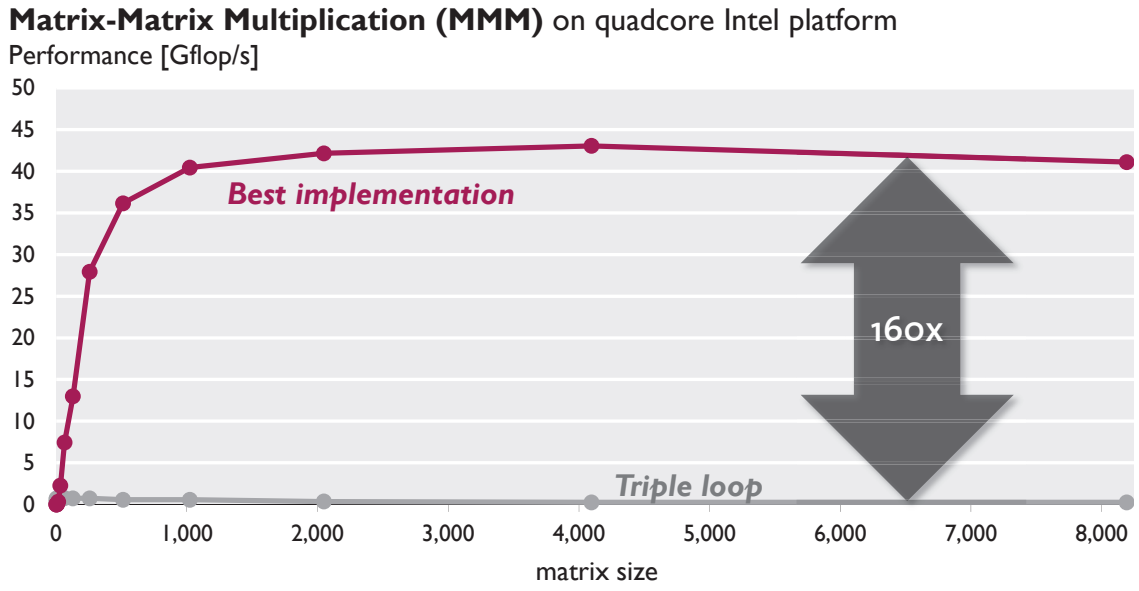
*with:*
Frédéric de Mesmay
PhD, Electrical and Computer Engineering, Carnegie Mellon

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SPIRAL
www.spiral.net

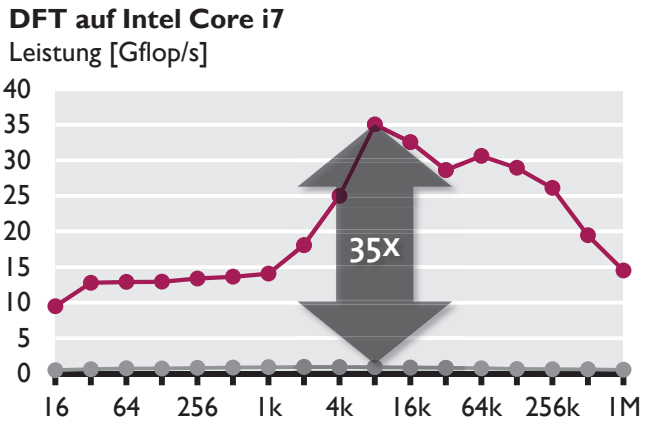


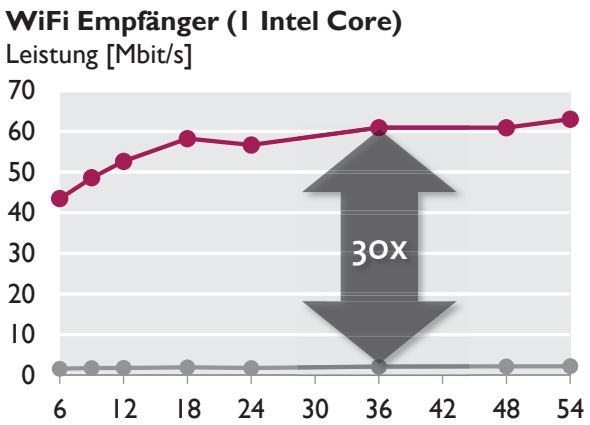
# Why Autotuning?




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

# Same for All Critical Compute Functions




Markus Püschel, ETH Zürich, 2011

# 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*

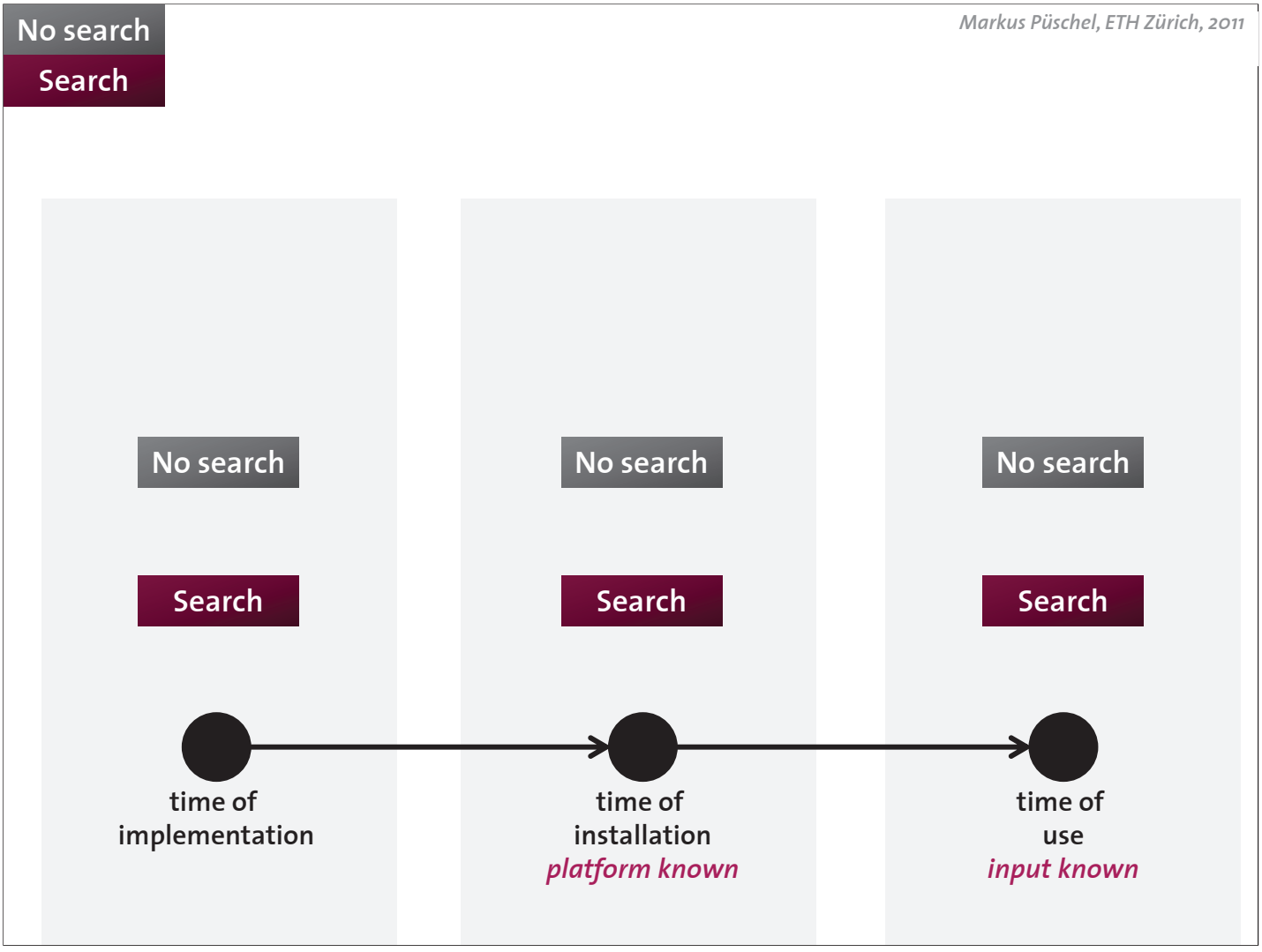
**However:** *Search is an important tool. But expensive.*

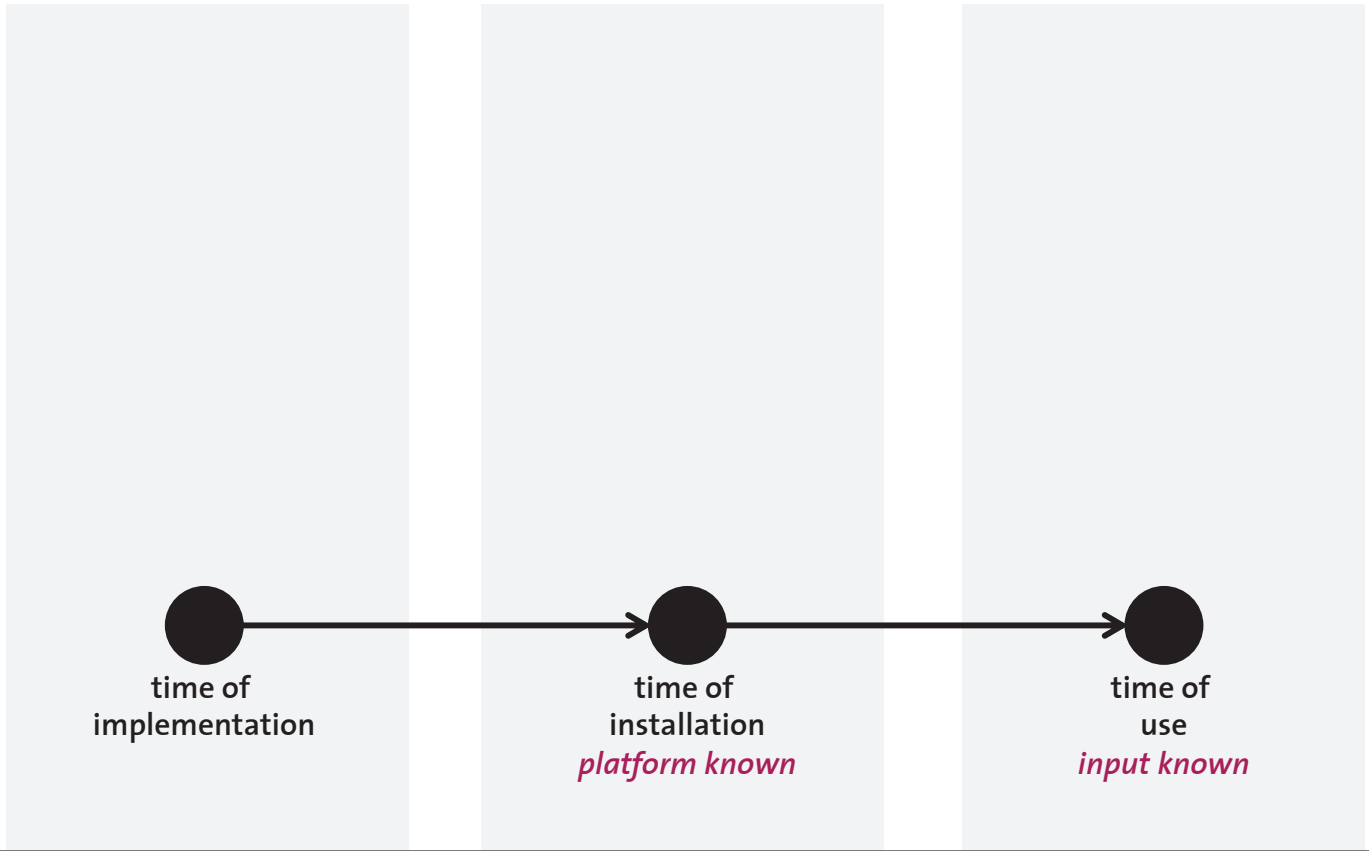
**Solution: Machine learning**

# Organization

- Autotuning examples
- An example use of machine learning










# PhiPac/ATLAS: MMM Generator

*Whaley, Bilmes, Demmel, Dongarra, ...*

```
// MMM loop-nest
for i = 0:N_B:N-1
   for j = 0:N_B:M-1
      for k = 0:N_B:K-1
```

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

```
// mini-MMM loop nest
for i' = i:M_U:i+N_B-1
   for j' = j:N_U:j+N_B-1
      for k' = k:K_U:k+N_B-1
```

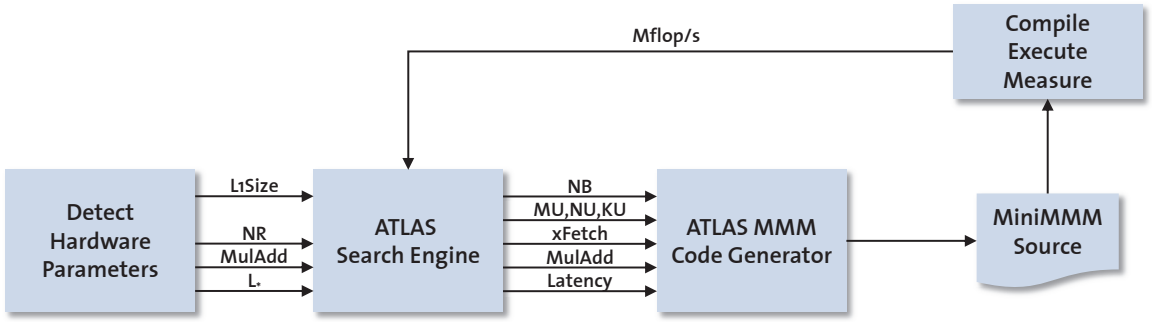
- *Blocking for registers*

```
// micro-MMM loop nest
for k" = k':1:k'+K_U-1
   for i" = i':1:i'+M_U-1
      for j" = j':1:j'+N_U-1
```

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

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

# PhiPac/ATLAS: MMM Generator

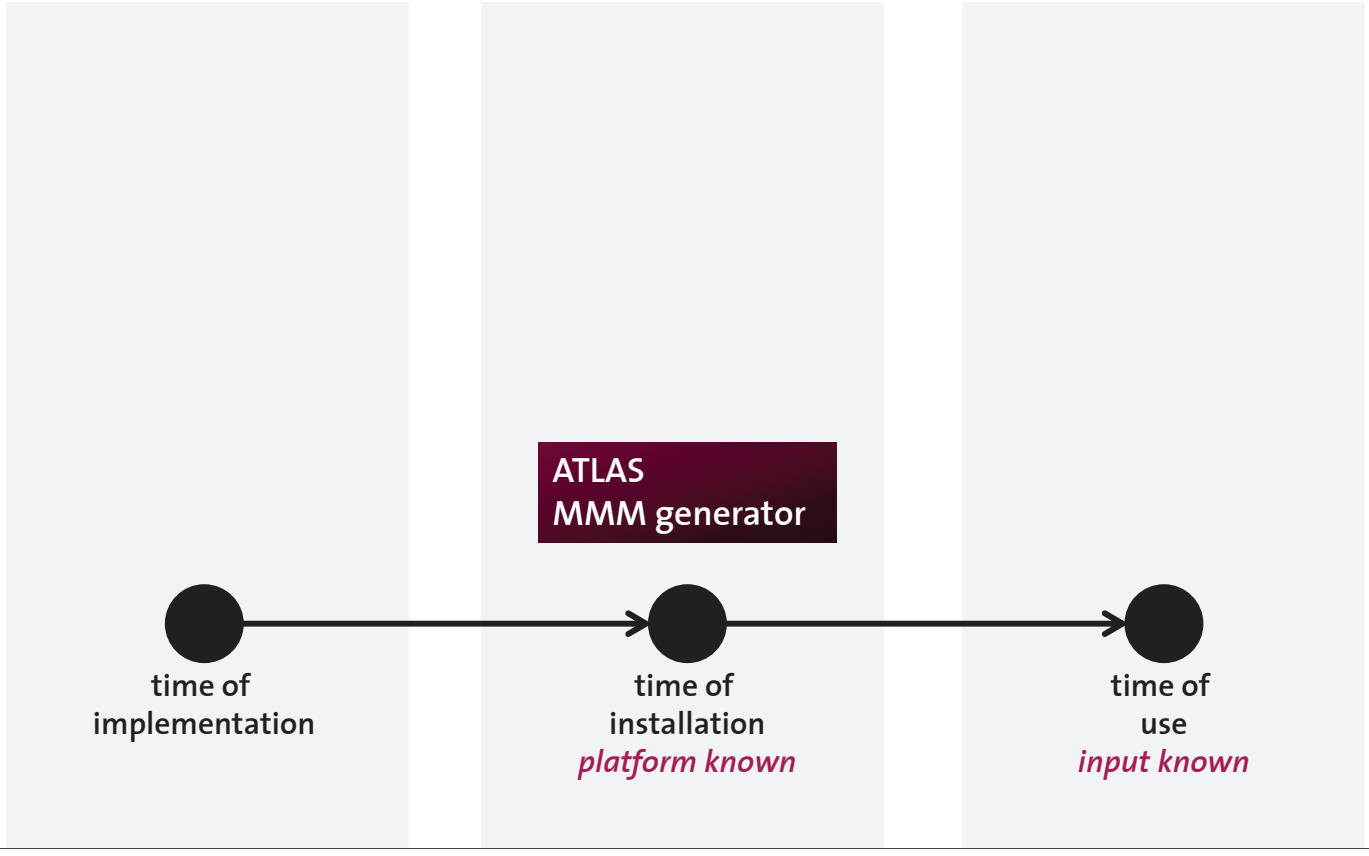



*source: Pingali, Yotov, Cornell U.*



No search

Search



# FFTW: Discrete Fourier Transform (DFT)

*Frigo, Johnson*

## Installation

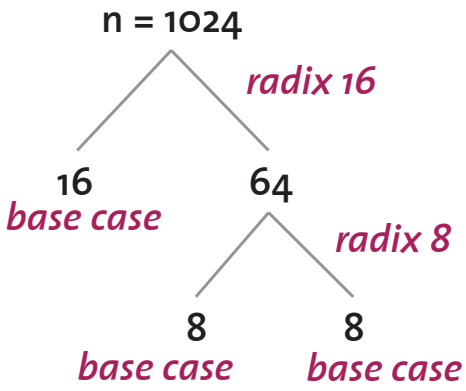
```
configure/make
```

## Usage

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

Twiddles

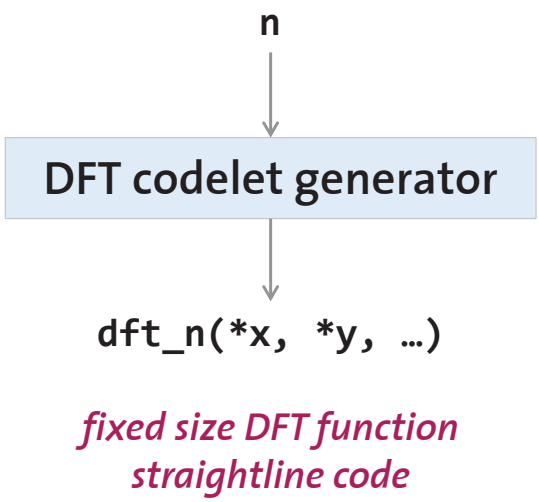
Search for fastest computation strategy



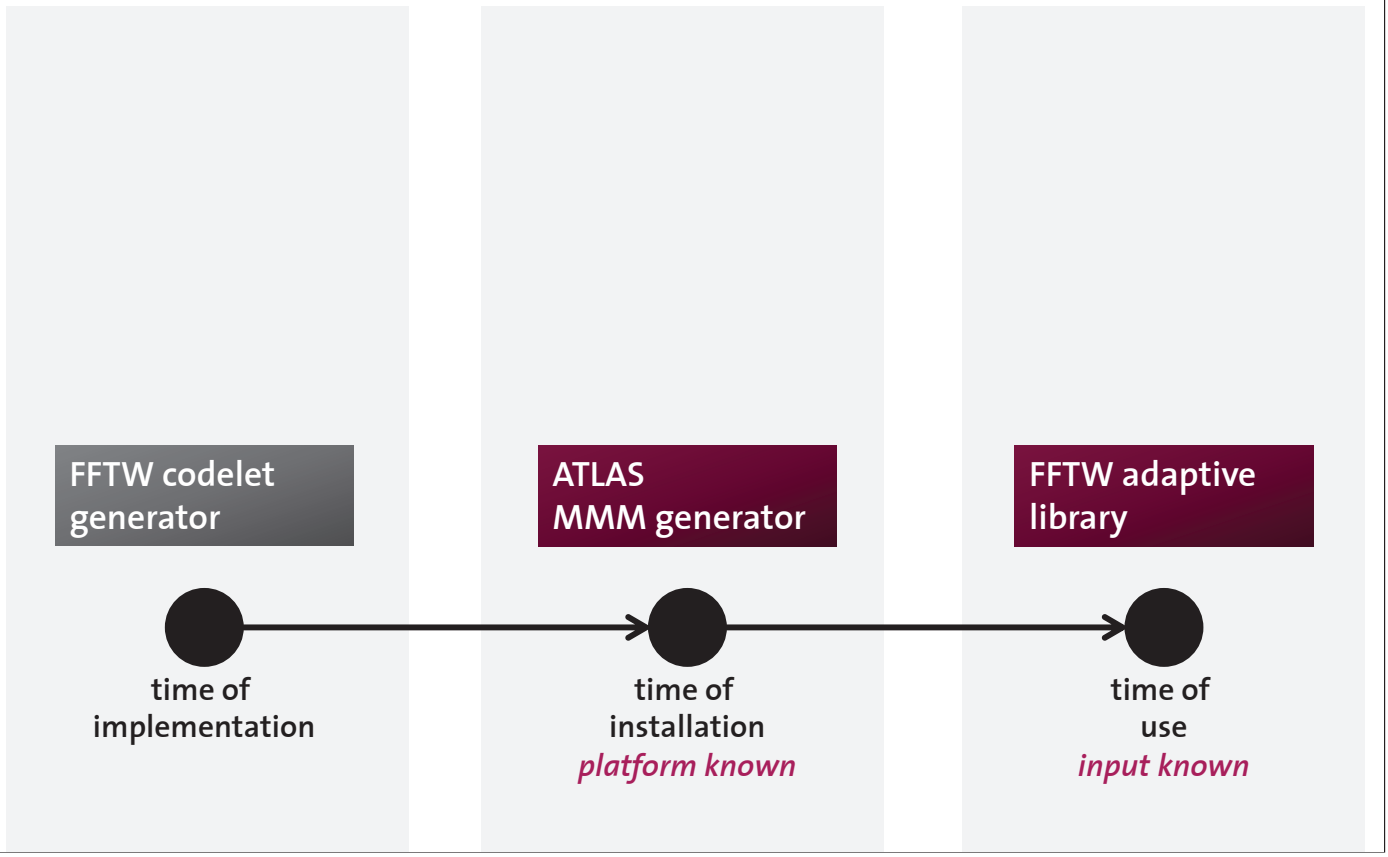



# FFTW: Codelet Generator

*Frigo*



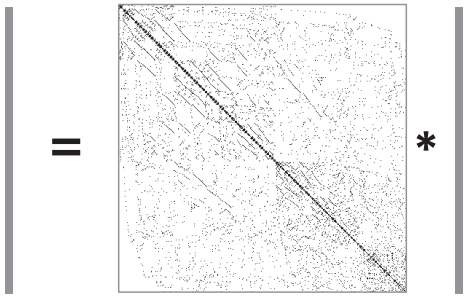

*fixed size DFT function*
*straightline code*



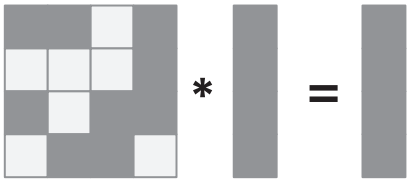



# OSKI: Sparse Matrix-Vector Multiplication

*Vuduc, Im, Yelick, Demmel*

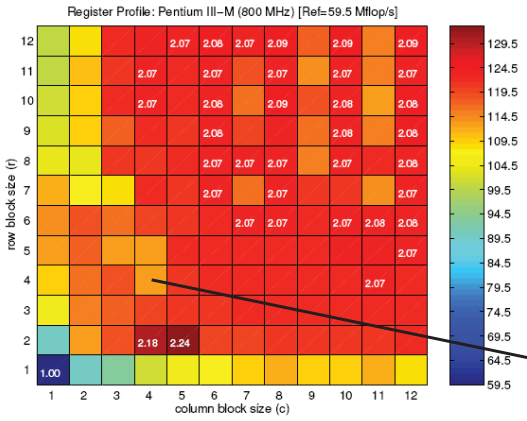



- **Blocking for registers:**
  - Improves locality (reuse of input vector)
  - But creates overhead (zeros in block)



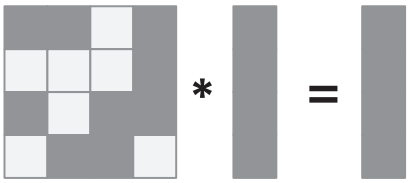
# OSKI: Sparse Matrix-Vector Multiplication

## Gain by blocking (dense MVM)




1.4

## Overhead by blocking



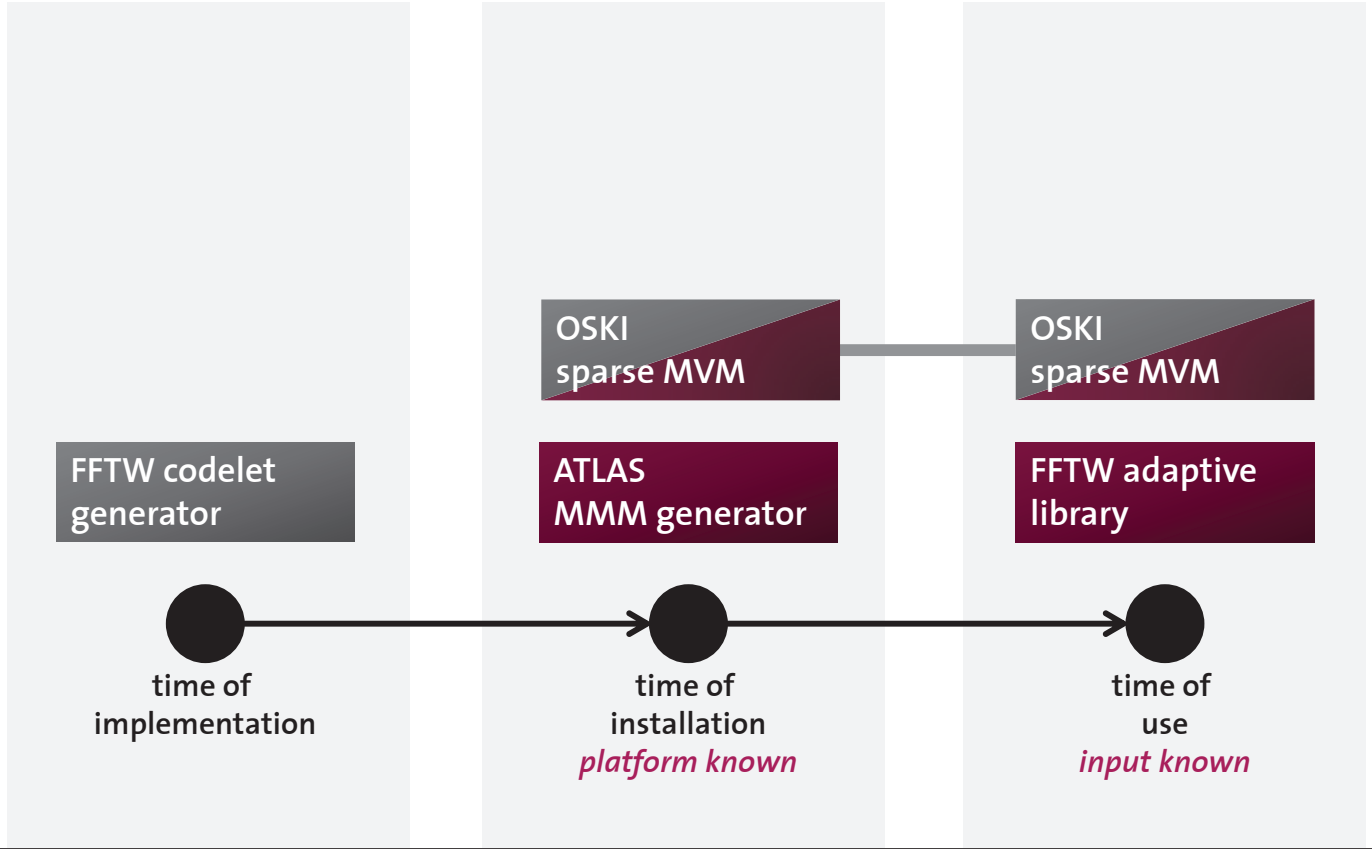
16/9 = 1.77

1.4/1.77 = 0.79 (no gain)



No search

Search



# Spiral: Linear Transforms & More

## Algorithm knowledge

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

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

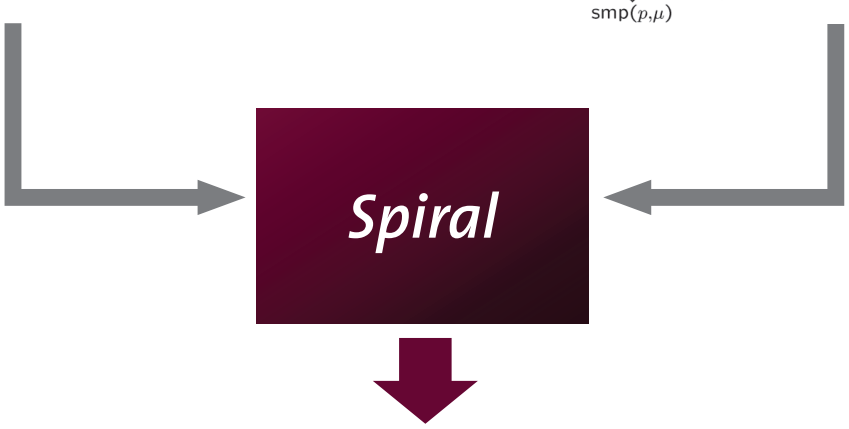
$$\mathbf{RDFT\text{-}3}_n \to (Q^{\top}_{k/2,m} \otimes I_2)\,(I_k \otimes_i \mathbf{rDFT}_{2m})(i+1/2)/k))\,(\mathbf{RDFT\text{-}3}_k \otimes I_m)$$

## Platform description

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

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

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




**Optimized implementation**

*regenerated for every new platform*



# Program Generation in Spiral (Sketched)

**Transform**
*user specified*

$$\mathsf{DFT}_8$$

*Algorithm rules*

**Fast algorithm in SPL**
*many choices*

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

**Σ-SPL**

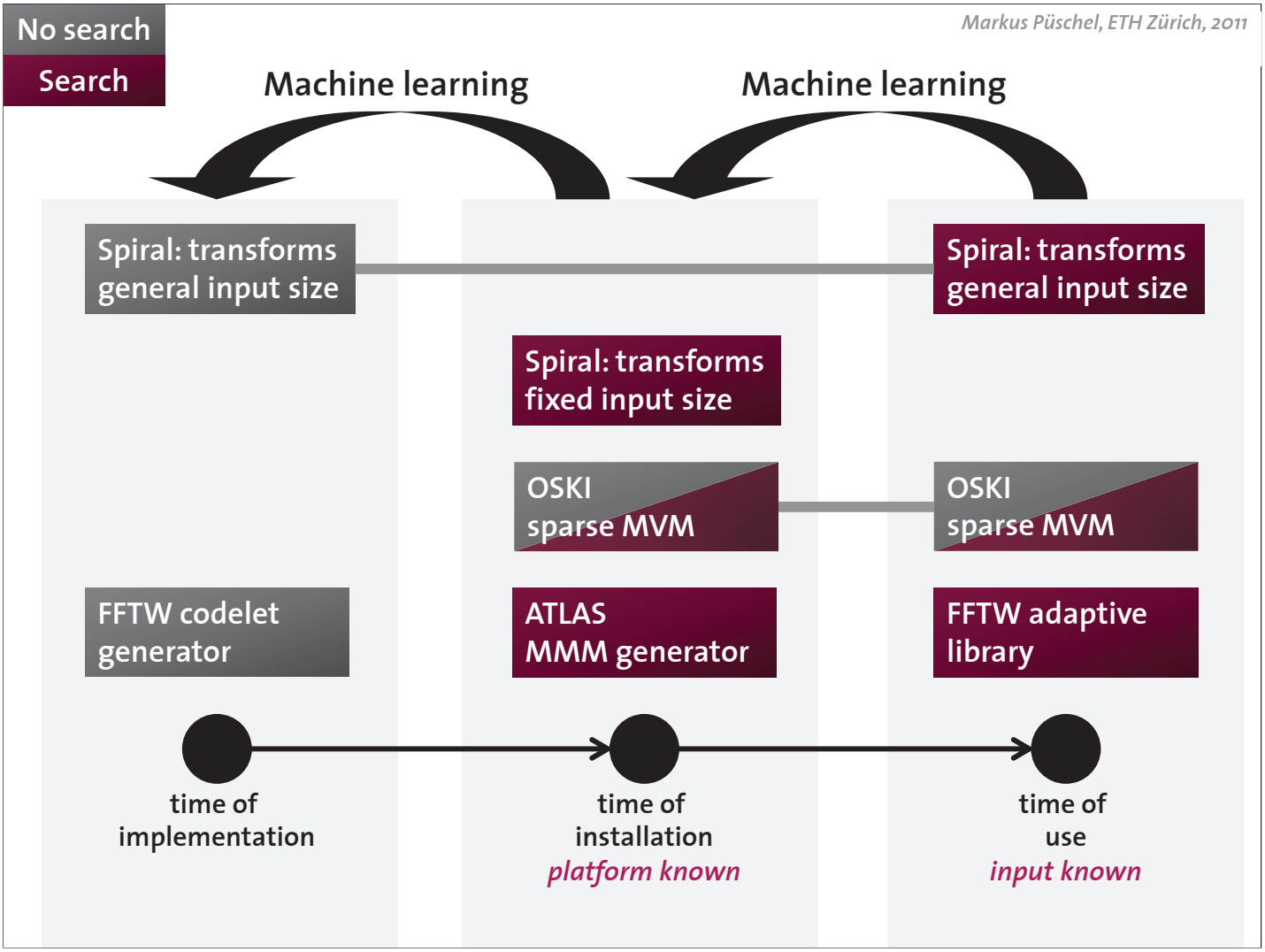
$$\sum\left(S_j\, \mathsf{DFT}_2\, G_j\right) \sum\left(\sum\left(S_{k,l}\, \mathsf{diag}\left(\mathrm{t}_{k,l}\right) \mathsf{DFT}_2\, G_l\right) \sum\left(S_m\, \mathsf{diag}\left(\mathrm{t}_m\right) \mathsf{DFT}_2\, G_{k,m}\right)\right)$$

**Optimized implementation**

***Optimization at all abstraction levels***

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

**+ search**




# Organization

- Autotuning examples
- An example use of machine learning

## *Online tuning (time of use)*

**Installation**

`configure/make`

**Use**

`d = dft(n)` — Twiddles; **Search for fastest computation strategy**

`d(x,y)`

## *Offline tuning (time of installation)*

**Installation**

`configure/make`

**for a few n: search learn decision trees**

**Use**

`d = dft(n)` — Twiddles

`d(x,y)`

**Goal**

Markus Püschel, ETH Zürich, 2011

# Integration with Spiral-Generated Libraries

*Voronenko 2008*

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

→ **Spiral** →

Online tunable library

$$
\begin{aligned}
\mathbf{DFT}_n &\to P^\top_{k/2,2m} \left(\mathbf{DFT}_{2m} \oplus \left(I_{k/2-1} \quad {}_i C_{2m}\, \mathbf{rDFT}_{2m}(i/k)\right)\right) \left(\mathbf{RDFT}'_k \quad I_m\right), \quad k \text{ even},\\
\begin{vmatrix}\mathbf{RDFT}_n\\ \mathbf{RDFT}'_n\\ \mathbf{DHT}_n\\ \mathbf{DHT}'_n\end{vmatrix} &\to (P^\top_{k/2,m} \quad I_2) \left(\begin{vmatrix}\mathbf{RDFT}_{2m}\\ \mathbf{RDFT}'_{2m}\\ \mathbf{DHT}_{2m}\\ \mathbf{DHT}'_{2m}\end{vmatrix} \oplus \left(I_{k/2-1} \quad {}_i D_{2m} \begin{vmatrix}\mathbf{rDFT}_{2m}(i/k)\\ \mathbf{rDFT}_{2m}(i/k)\\ \mathbf{rDHT}_{2m}(i/k)\\ \mathbf{rDHT}_{2m}(i/k)\end{vmatrix}\right)\right) \left(\begin{vmatrix}\mathbf{RDFT}'_k\\ \mathbf{RDFT}'_k\\ \mathbf{DHT}'_k\\ \mathbf{DHT}'_k\end{vmatrix} \quad I_m\right), \quad k \text{ even},\\
\begin{vmatrix}\mathbf{rDFT}_{2n}(u)\\ \mathbf{rDHT}_{2n}(u)\end{vmatrix} &\to L^{2n}_m \left(I_k \quad {}_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} \quad I_m\right),\\
\mathbf{RDFT\text{-}3}_n &\to (Q^\top_{k/2,m} \quad I_2)\, (I_k \quad {}_i\, \mathbf{rDFT}_{2m})(i+1/2)/k))\, (\mathbf{RDFT\text{-}3}_k \quad 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} \quad N_{2m}\, \mathbf{RDFT\text{-}3}^\top_{2m}\right)\right) B_n (L^{n/2}_{k/2} \quad I_2)(I_m \quad \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} \quad N_{2m}\, \mathbf{RDFT\text{-}3}^\top_{2m}\right) B'_n (L^{n/2}_{k/2} \quad I_2)(I_m \quad \mathbf{RDFT\text{-}3}_k) Q_{m/2,k}.\\
\mathbf{DFT}_n &\to (\mathbf{DFT}_k \quad \mathrm{I}_m)\, \mathsf{T}^n_m (\mathrm{I}_k \quad \mathbf{DFT}_m)\, \mathsf{L}^n_k, \quad n = km\\
\mathbf{DFT}_n &\to P_n (\mathbf{DFT}_k \quad \mathbf{DFT}_m) Q_n, \quad n = km,\ \gcd(k,m) = 1\\
\mathbf{DFT}_p &\to R_p^T (\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)\, \mathsf{L}^n_m (\mathbf{DCT\text{-}3}_m(1/4) \oplus \mathbf{DCT\text{-}3}_m(3/4))\\
&\qquad \cdot (\mathrm{F}_2 \quad \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} \quad \mathrm{I}_m\right) \oplus \left(\begin{bmatrix}-1\\ -1\end{bmatrix} \quad \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}}} \quad \mathbf{WHT}_{2^{k_i}} \quad \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

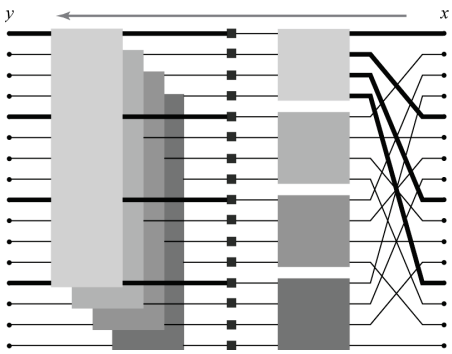
- Discrete Fourier transform (DFT):

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

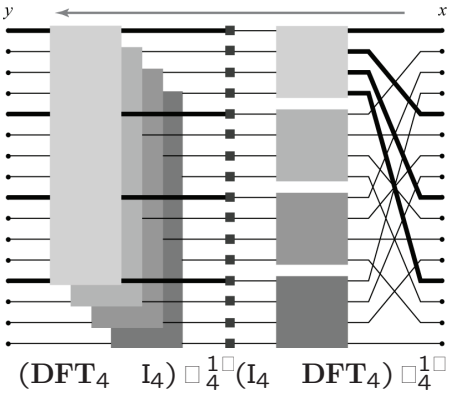
- Cooley/Tukey fast Fourier transform (FFT):

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

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



# Fast Fourier Transform




16 = 4 x 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) { ... }
```

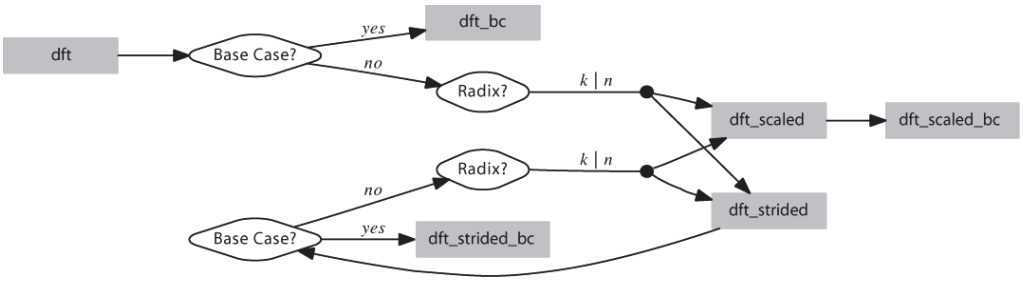
Choices used for adaptation



# Decision Graph of Library

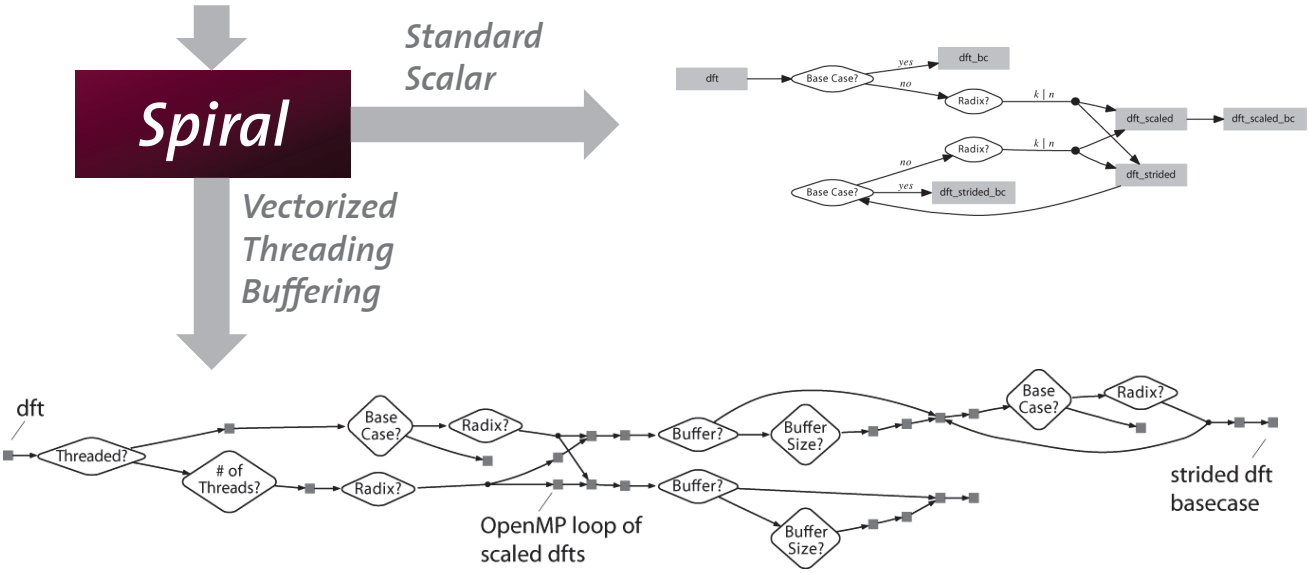
```
void dft(int n, cpx *y, cpx *x) {
    if (use_dft_base_case(n))
        dft_bc(n, y, x);
    else {
        int k = choose_dft_radix(n);
        for (int i=0; i < k; ++i)
            dft_strided(m, k, t + m*i, x + m*i);
        for (int i=0; i < m; ++i)
            dft_scaled(k, m, precomp_d[i],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

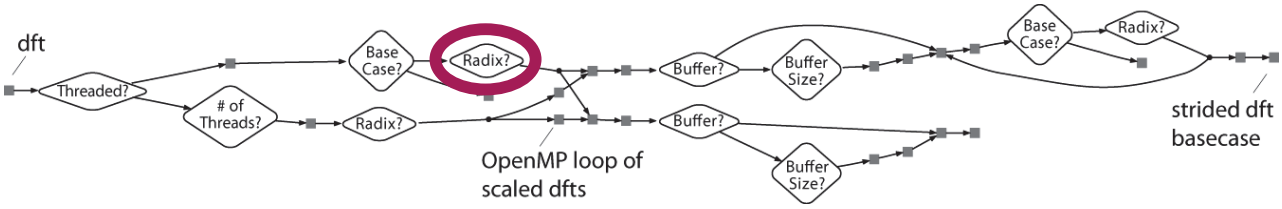
$$(\mathrm{DFT}_k \otimes \mathrm{I}_m)\, \mathrm{T}^n_m (\mathrm{I}_k \otimes \mathrm{DFT}_m)\, \mathrm{L}^n_k$$




- 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:*

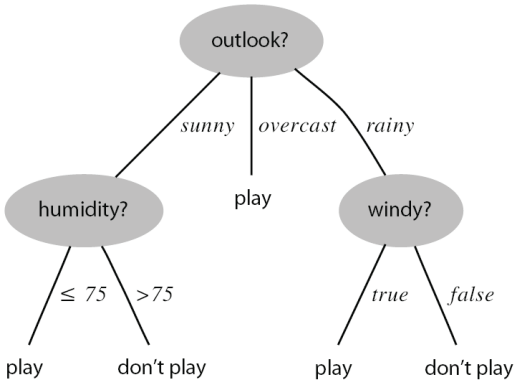
```
if ( n <= 65536 ) {
   if ( n <= 32 ) {
      if ( n <= 4 ) {return 2;}
      else {return 4;}
   }
   else {
      if ( n <= 1024 ) {
         if ( n <= 256 ) {return 8;}
         else {return 32;}
      }
      else {
      ••••••••••••••••••
```

# 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 |

C4.5




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

*Entropy of Features*

H(windy) = 0.89
**H(outlook) = 0.69**
H(humidity) = ...



# Application to Libraries

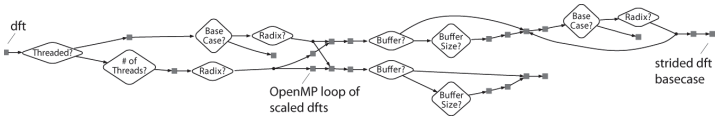
- **Features = arguments of functions (except variable pointers)**

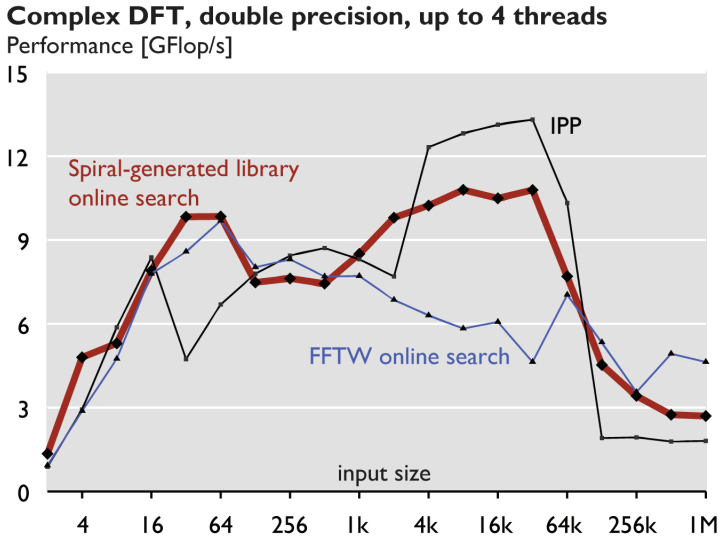
```
dft(int n, cpx *y, cpx *x)
dft_strided(int n, int istr, cpx *y, cpx *x)
dft_scaled(int n, int str, cpx *d, cpx *y, cpx *x)
```

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

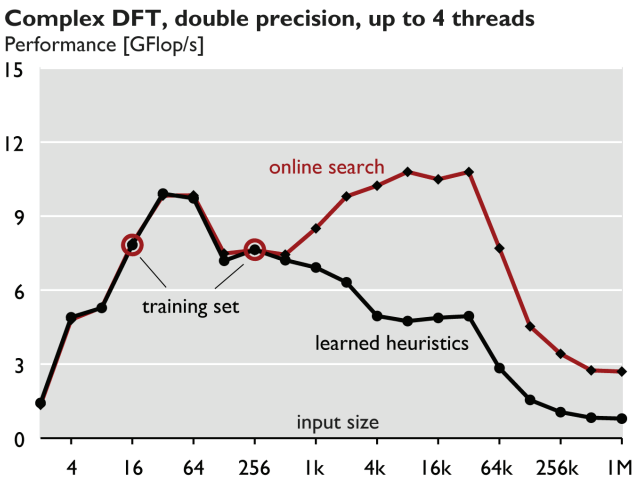
# Experimental Setup

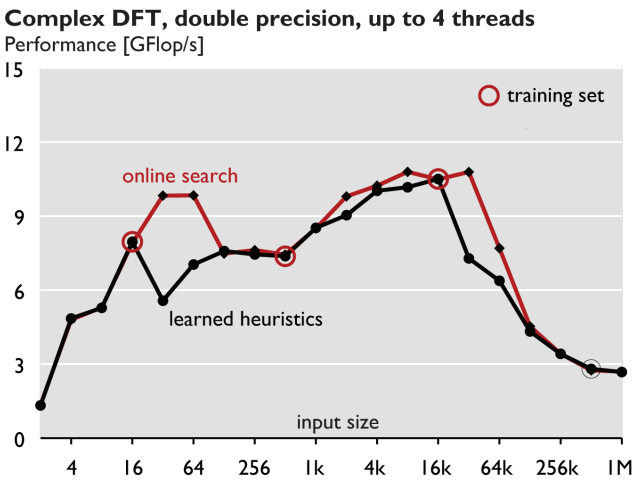
- 3GHz Intel Xeon 5160 (2 Core 2 Duos = 4 cores), icc 10.1
- Spiral-generated library:

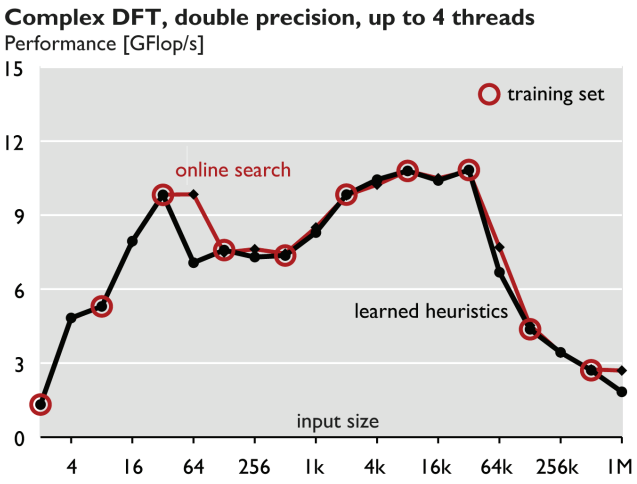



Markus Püschel, ETH Zürich, 2011

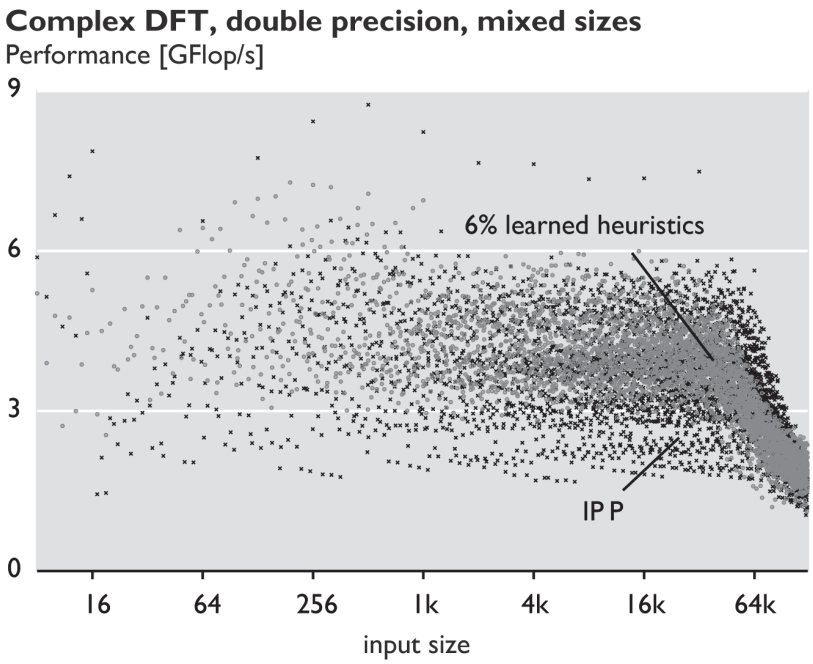



Learning works as expected

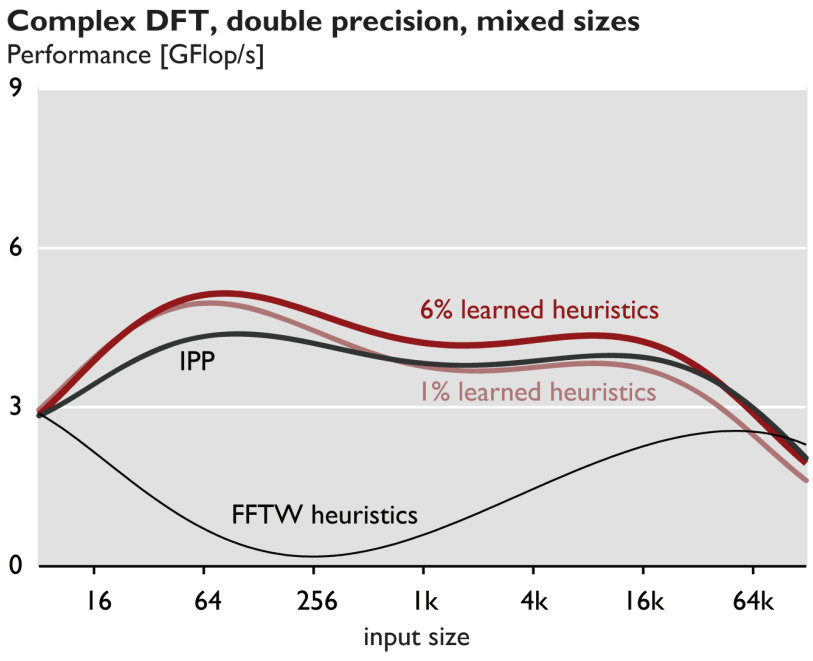
# "All" Sizes




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

Markus Püschel, ETH Zürich, 2011

# "All" Sizes




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

Markus Püschel, ETH Zürich, 2011

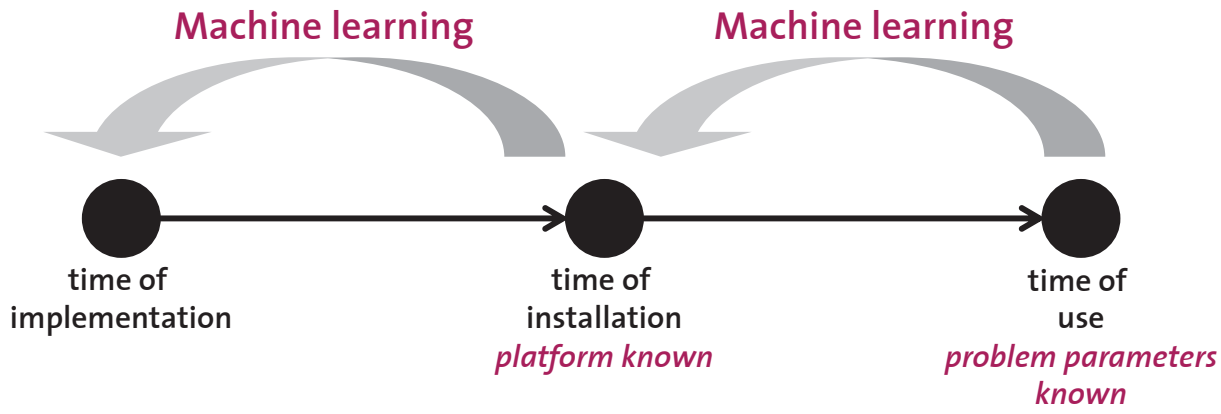
# This Talk

- **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 ≠ search**