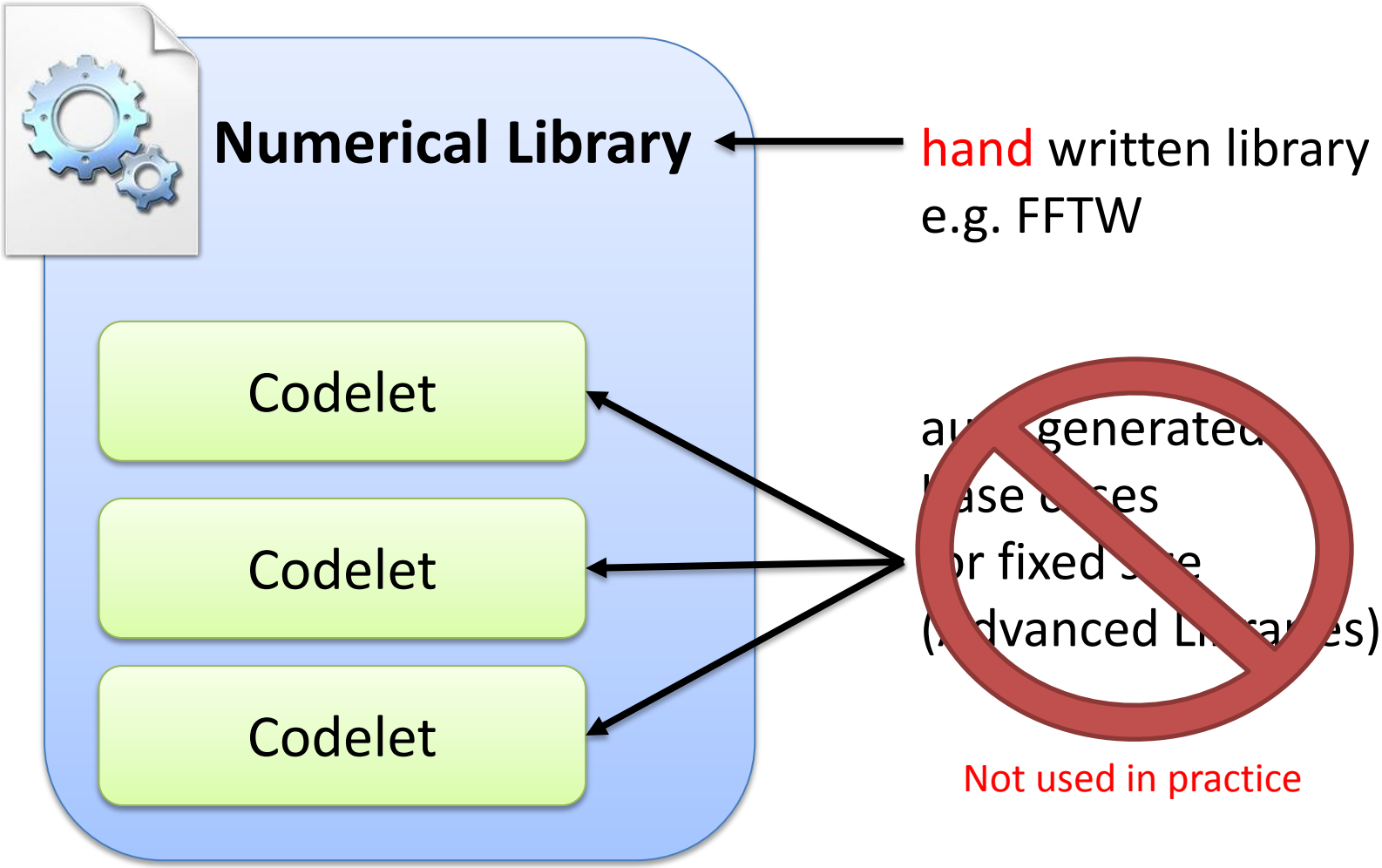
Computer Generation of General Size Linear Transform Libraries

Yevgen Voronenko, Frédéric de Mesmay, and Markus Püschel
Carnegie Mellon University, Pittsburgh, USA

PROBLEM AND MOTIVATION

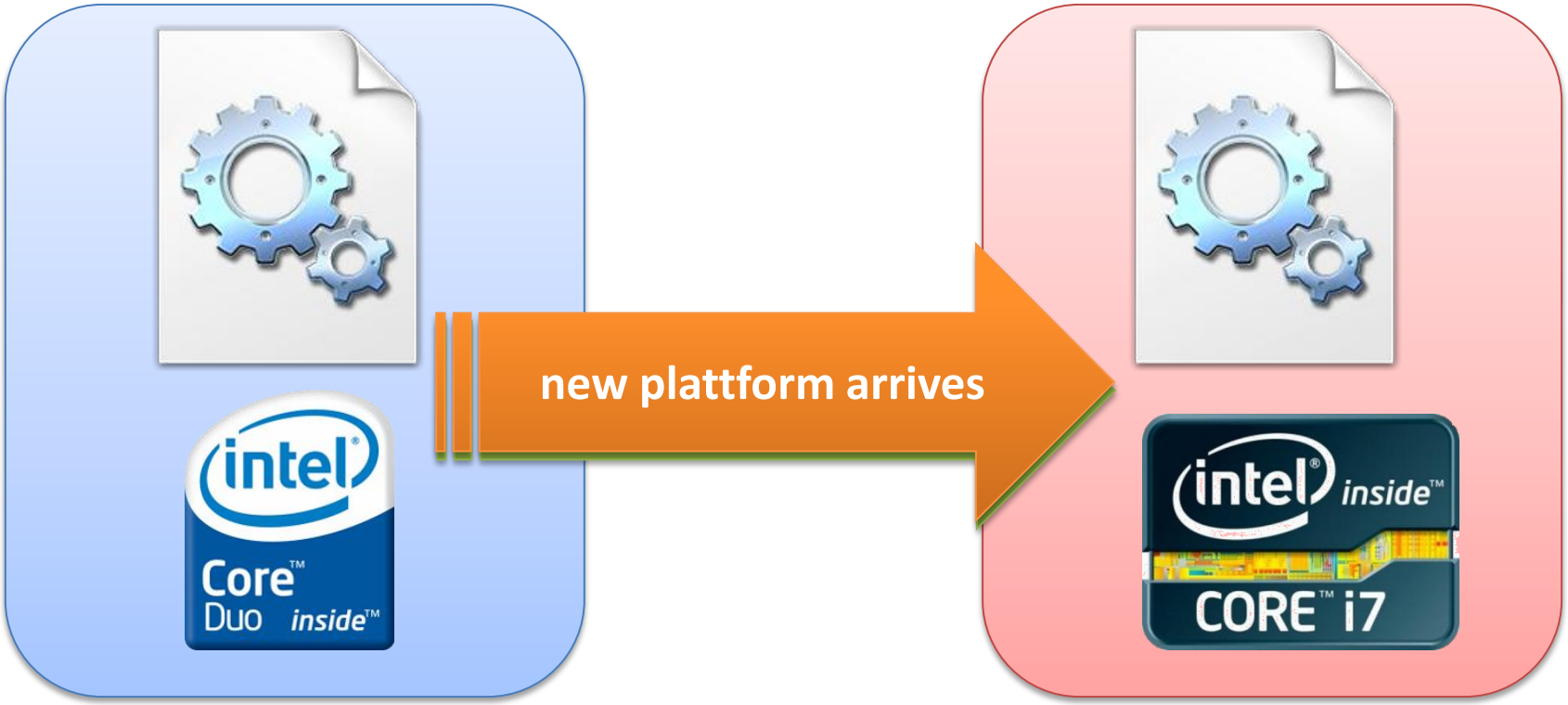
http://4.bp.blogspot.com/_LkcgSIEg0jw/TOi0XygeC4I/AAAAAAAAAH0/K0FyPn28gtc/s1600/Confuse.jpeg

Situation Today



http://en.wikipedia.org/wiki/File:DLL_icon_on_Windows_Vista.png

Problem

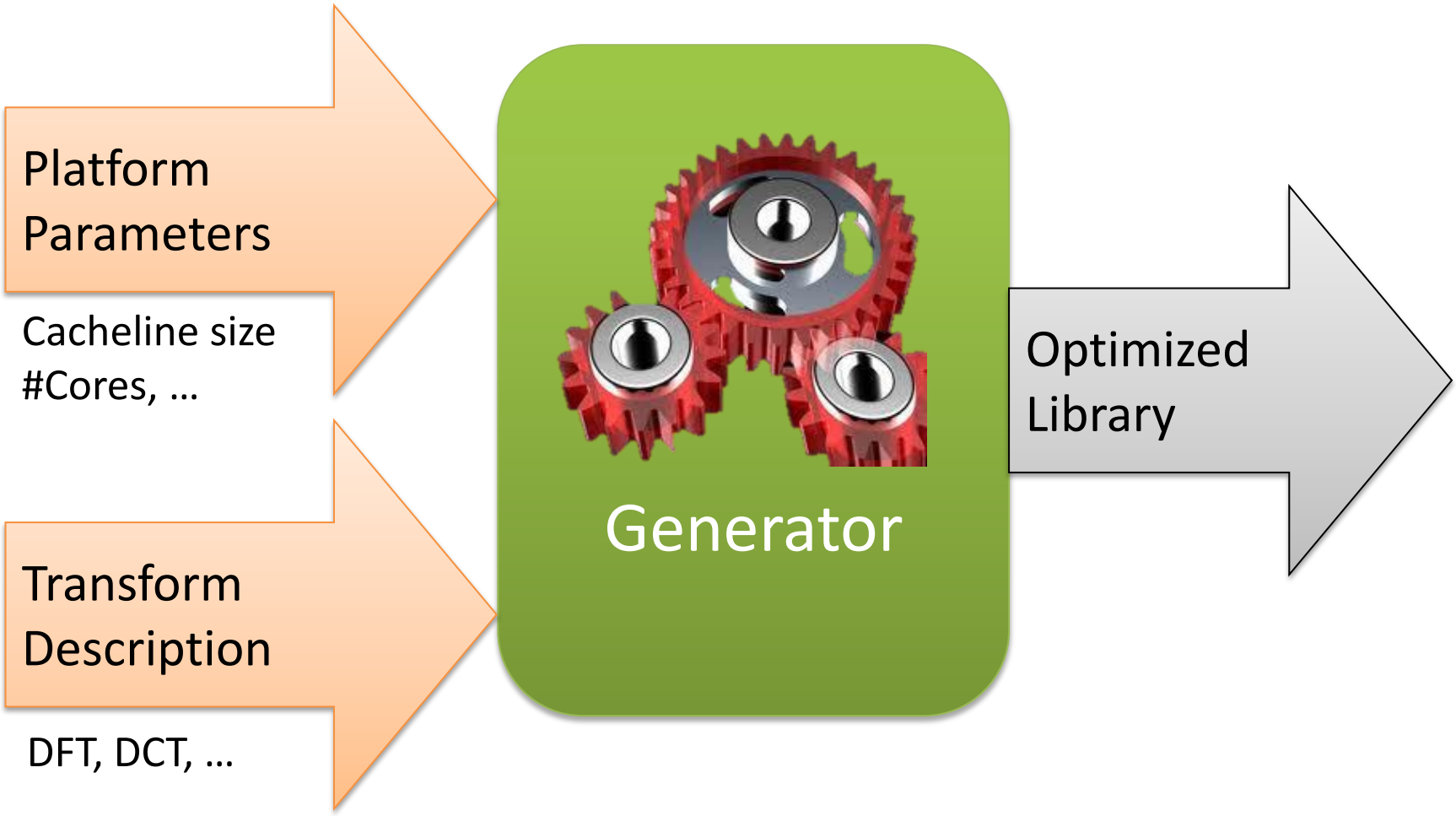


optimize & implement
for spezific plattform

reoptimize & reimplement
on new plattform

http://en.wikipedia.org/wiki/File:DLL_icon_on_Windows_Vista.png
http://en.wikipedia.org/wiki/Intel

Goal



http://www.anilgears.com/images1/gear6.jpg

BACKGROUND ON LINEAR TRANSFORMS

http://www.dimelo.com/img/style/analytics.jpg

Example DFT

Discrete Fourier Transform (DFT): $y = DFT_n \cdot x$

$$\boldsymbol{DFT_n} = (\boldsymbol{DFT_k} \otimes I_m)\, T^n_m\, (I_k \otimes \boldsymbol{DFT_m})\, L^n_k$$

base case: $DFT_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Recursive Cases

Example: Implementation (Naïve)

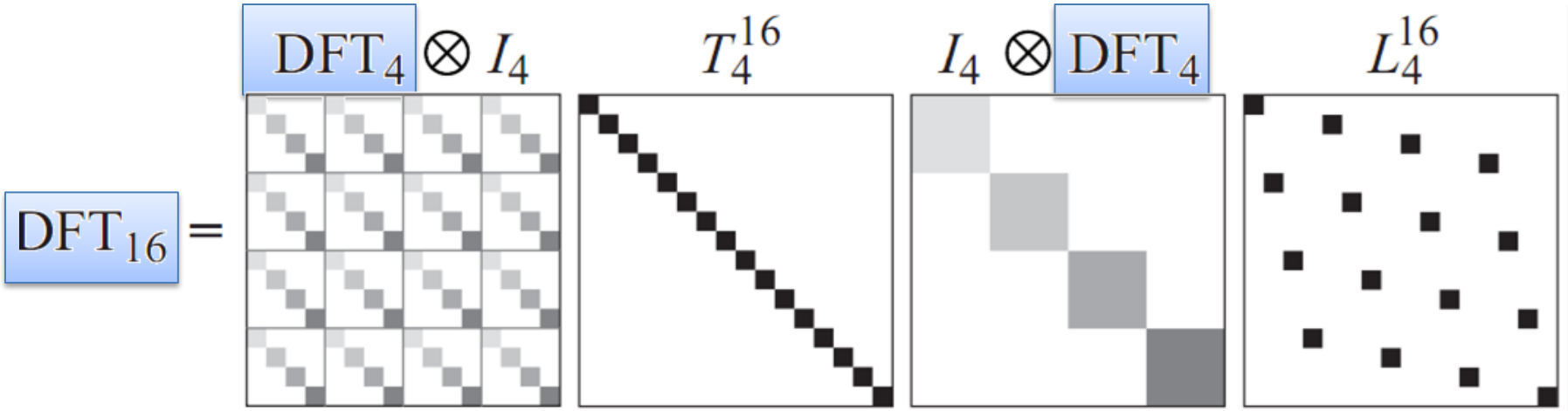
$$DFT_n = (DFT_k \otimes I_m)\, T^n_m\, (I_k \otimes DFT_m)\, L^n_k$$

```
void dft(int n, cpx *y, cpx *x) {
    int k = choose_factor(n);
    int m = n/k;
    cpx *t1 = Permute x with L(n,k);
    // t2 = (I_k tensor DFT_m)*t1
    for(int i=0; i<k; ++i)
        dft(m, t2 + m*i, t1 + m*i);
    // t3 = diag( d(j) )*t2
    for(int i=0; i<n; ++i)
        t3[i] = d(i) * t2[i];
    // y = (DFT_k tensor I_m)*t3,
    for(int i=0; i<m; ++i)
        dft_str(k, m, y + i, t3 + i);
}
// to be implemented
void dft_str(int n, int str, cpx *Y, cpx *X);
```

memory to memory copy **SLOW!**

DFT – Matrix Representation

$$\boldsymbol{DFT_n} = (\boldsymbol{DFT_k} \otimes I_m)\, T^n_m\, (I_k \otimes \boldsymbol{DFT_m})\, L^n_k$$



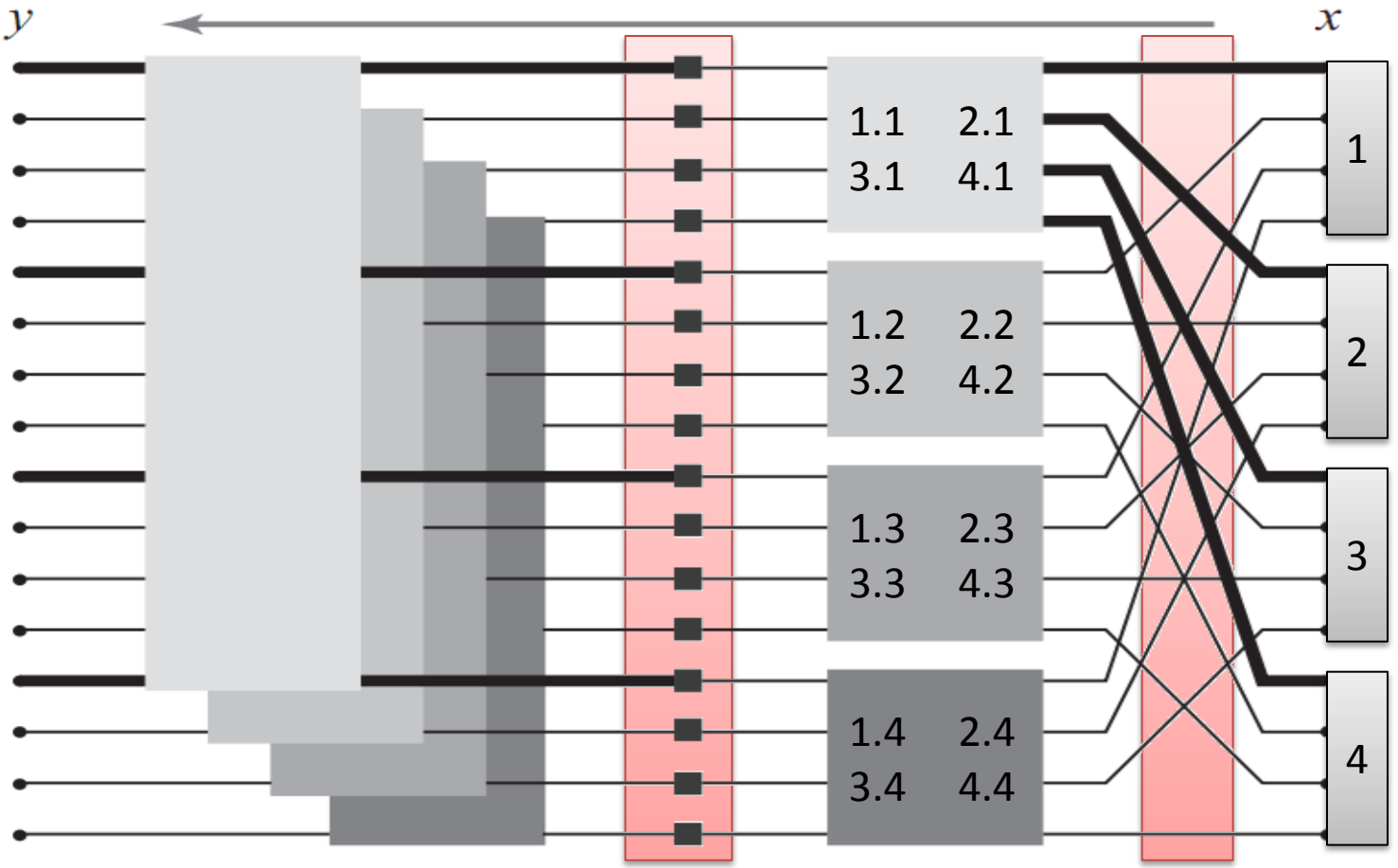
Diagonal Matrix

Input Permutation

Recursive Cases

graphics from paper

DFT – Dataflow Representation



$$DFT_n = (DFT_k \otimes I_m)\; T^n_m\; (I_k \otimes DFT_m)\; L^n_k$$

DFT – Improvement Idea

$$\boldsymbol{DFT_n} = \boldsymbol{(DFT_k \otimes I_m)}\ T_m^n\ \boldsymbol{(I_k \otimes DFT_m)}\ L_k^n$$

T_m^n: A scalar multiplier for each entry

L_k^n: A permutation function

Merge

$$\boldsymbol{DFT_n} = \underbrace{\boldsymbol{(DFT_k \otimes I_m)}\ T_m^n}_{\text{“Scaled DFT”}}\ \underbrace{\boldsymbol{(I_k \otimes DFT_m)}\ L_k^n}_{\text{“Strided DFT”}}$$

Example – Implementation (FFTW)

$$\mathbf{DFT_n} = (\mathbf{DFT_k} \otimes \mathbf{I_m})\ \mathbf{T^n_m}\ (\mathbf{I_k} \otimes \mathbf{DFT_m})\ \mathbf{L^n_k}$$

```
void dft(int n, cpx *y, cpx *x) {
    int k = choose_factor(n);
    // t1 = (I_k tensor DFT_m)L(n,k)*x
    for(int i=0; i < k; ++i)
        dft_str(m, k, 1, t1 + m*i, x + m*i);
    // y = (DFT_k tensor I_m) diag(d(j))
    for(int i=0; i < m; ++i)
        dft_scaled(k, m, precomp_d[i], y + i, t1 + i);
}
// to be implemented
void dft_str(int n, int istr, int ostr, cpx *y, cpx*x);
void dft_scaled(int n, int str, cpx *d, cpx *y, cpx*x);
```

Challenge: How to find those recursive functions?

Example - Observation

Not yet parallelized and vectorized

Many other recursive functions needed

Optimizations are to be identified and implemented by hand

http://bookszeus.files.wordpress.com/2011/09/hands-on-keyboard.jpg

LIBRARY GENERATOR

http://www.artistsvalley.com/images/icons/Database%20Application%20Icons/Data%20Generator/256x256/Data%20Generator.jpg

Transform Description

Formulated in special language (SPL)

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

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

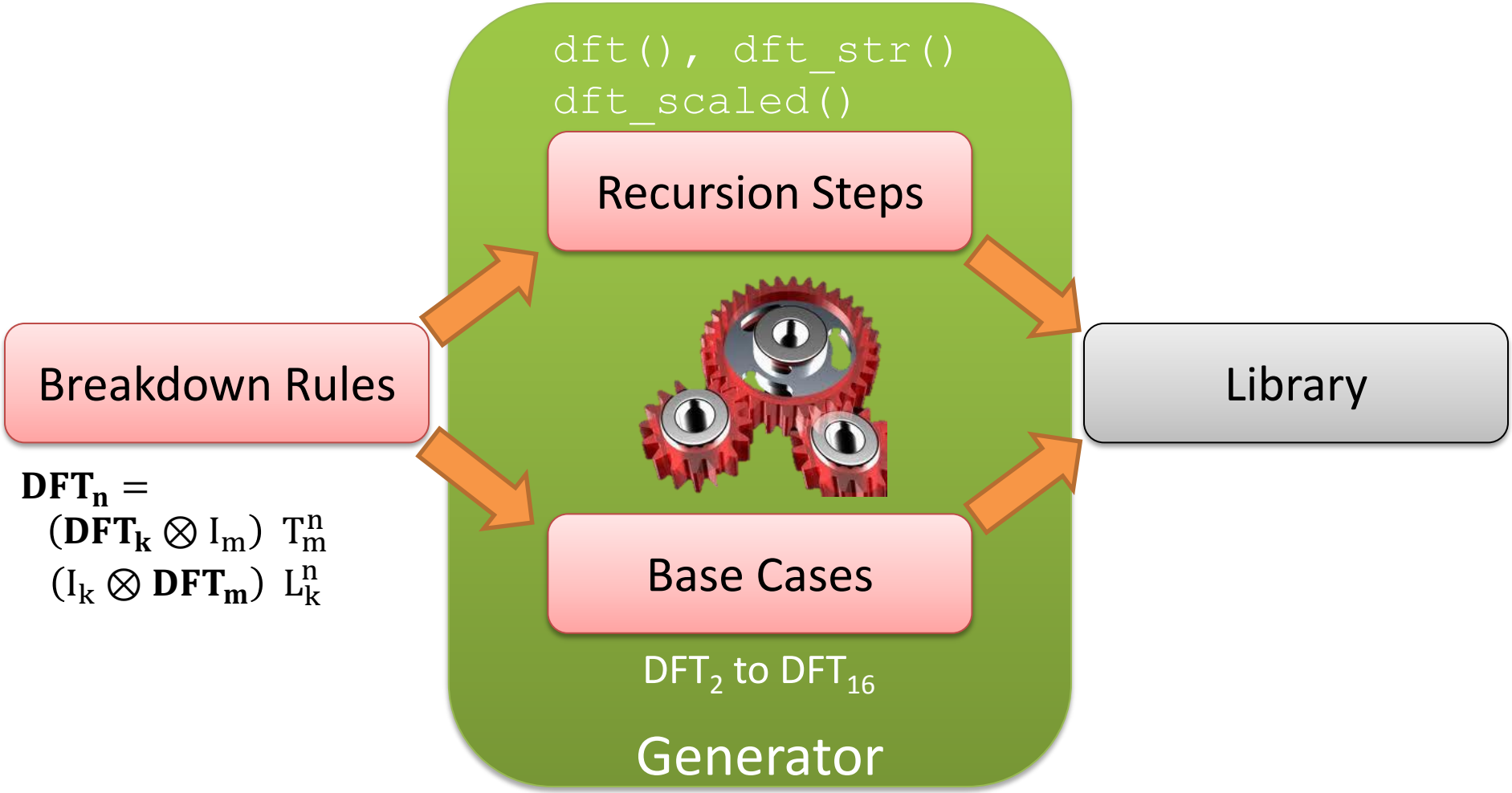
$$\mathbf{rDFT}_{2n}(u) = L^{2n}_m \left(I_k \otimes_i \mathbf{rDFT}_{2m}((i+u)/k)\right) \left(\mathbf{rDFT}_{2k}(u) \otimes I_m\right),$$

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

New set of breakdown rules for each transform

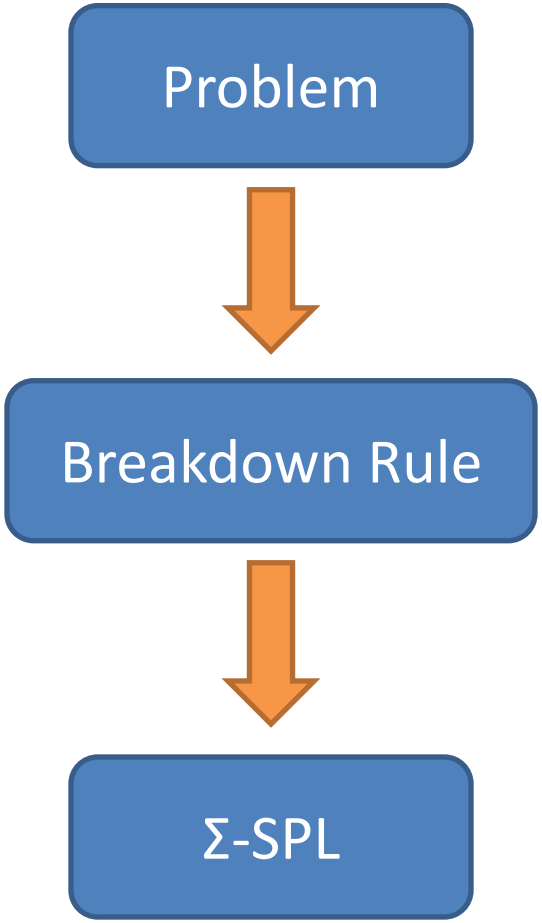
Can be complicated

Library Generation for Transforms



http://www.anilgears.com/images1/gear6.jpg

From SPL to Σ-SPL



A Linear Transform: $\mathbf{DFT_n}$

Breakdown rules in SPL:

$$\mathbf{DFT_n} = (\mathbf{DFT_k} \otimes I_m)\, T^n_m\, (I_k \otimes \mathbf{DFT_m})\, L^n_k$$

Rewritten breakdown rules into Σ-SPL

Difference of SPL and Σ-SPL

Σ-SPL = Extended SPL

+ Loop constructs **Σ**: iterative matrix sum

+ explicit I/O: gather *G(f)* and scatter *S(f)*

+ index mapping functions *M(f)*
for strides (permutations)

How to rewrite SPL to Σ-SPL?

Functions: $f: A \rightarrow B$

A: Integer interval [1..n]

B: Integer interval [1..n] or $\mathbb{C}$, $\mathbb{R}$

Example: Permutation $f^{n \rightarrow n}$ $(bijective)$

Σ-SPL – Gather and Scatter Matrices

Parameterized matrices

$$G(f) := \left[e_{f(0)}, \ldots, e_{f(n-1)}\right]^T = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$$

$$S(f) := G(f)^T = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$$

$$y = G(f) \cdot x \Leftrightarrow y_i = x_{f(i)}$$
$$y = S(f) \cdot x \Leftrightarrow y_{f(i)} = x_i$$

Inside the Tensor Product

$$\mathbf{DFT_n} = (\mathbf{DFT_k} \otimes \mathrm{I_m})\ \mathrm{T^n_m}\ \ (\mathrm{I_k} \otimes \mathbf{DFT_m})\ \ \mathrm{L^n_k}$$

$$\mathrm{Y} = (I_k \otimes A)\mathrm{x}$$

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$$

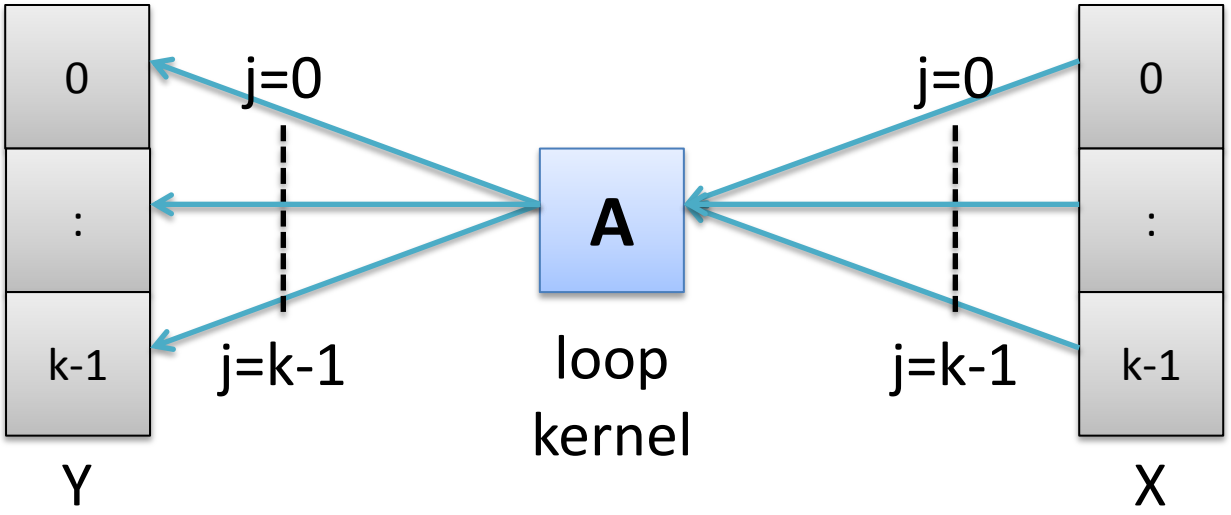
$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \left(\begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} + \cdots + \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix} \right) \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$$

Iterative Matrix Sum

Partition of vectors: $Y_i = A \cdot X_i$

Inside the Tensor Product

$$\mathbf{DFT_n} = (\mathbf{DFT_k} \otimes \mathrm{I_m})\ \mathrm{T^n_m}\ \ (\mathrm{I_k} \otimes \mathbf{DFT_m})\ \ \mathrm{L^n_k}$$



$$(I_k \otimes A) = S_{0A} G_0 + \cdots + S_{k-1} A\ G_{k-1} = \sum_{j=0}^{k-1} S_j A\ G_j$$

Σ-SPL – Rewriting Rules (Extract)

SPL to Σ-SPL

$$
\begin{aligned}
A \otimes I_k &\rightarrow \sum_{j=0}^{\kappa-1} \mathrm{S}(h_{j,k})\, A\, \mathrm{G}(h_{j,k}) \\
I_k \otimes A &\rightarrow \sum_{j=0}^{k-1} \mathrm{S}(h_{mj,1})\, A\, \mathrm{G}(h_{mj,1}) \\
L_k^n &\rightarrow \mathrm{perm}(\ell_k^n)
\end{aligned}
$$

Loop merging

$$
\begin{aligned}
\mathrm{G}(f)\,\mathrm{G}(g) &\rightarrow \mathrm{G}(g \circ f) \\
\mathrm{S}(f)\,\mathrm{S}(g) &\rightarrow \mathrm{S}(f \circ g) \\
\mathrm{G}(f)\,\mathrm{perm}(g) &\rightarrow \mathrm{G}(g \circ f) \\
\mathrm{perm}(f)\,\mathrm{S}(g) &\rightarrow \mathrm{S}(f^{-1} \circ g) \\
\mathrm{G}(f)\,\mathrm{diag}\,(d) &\rightarrow \mathrm{diag}\,(d \circ f)\,\mathrm{G}(f) \\
\mathrm{diag}\,(d)\,\mathrm{S}(f) &\rightarrow \mathrm{S}(f)\,\mathrm{diag}\,(d \circ f)
\end{aligned}
$$

From SPL to Σ-SPL

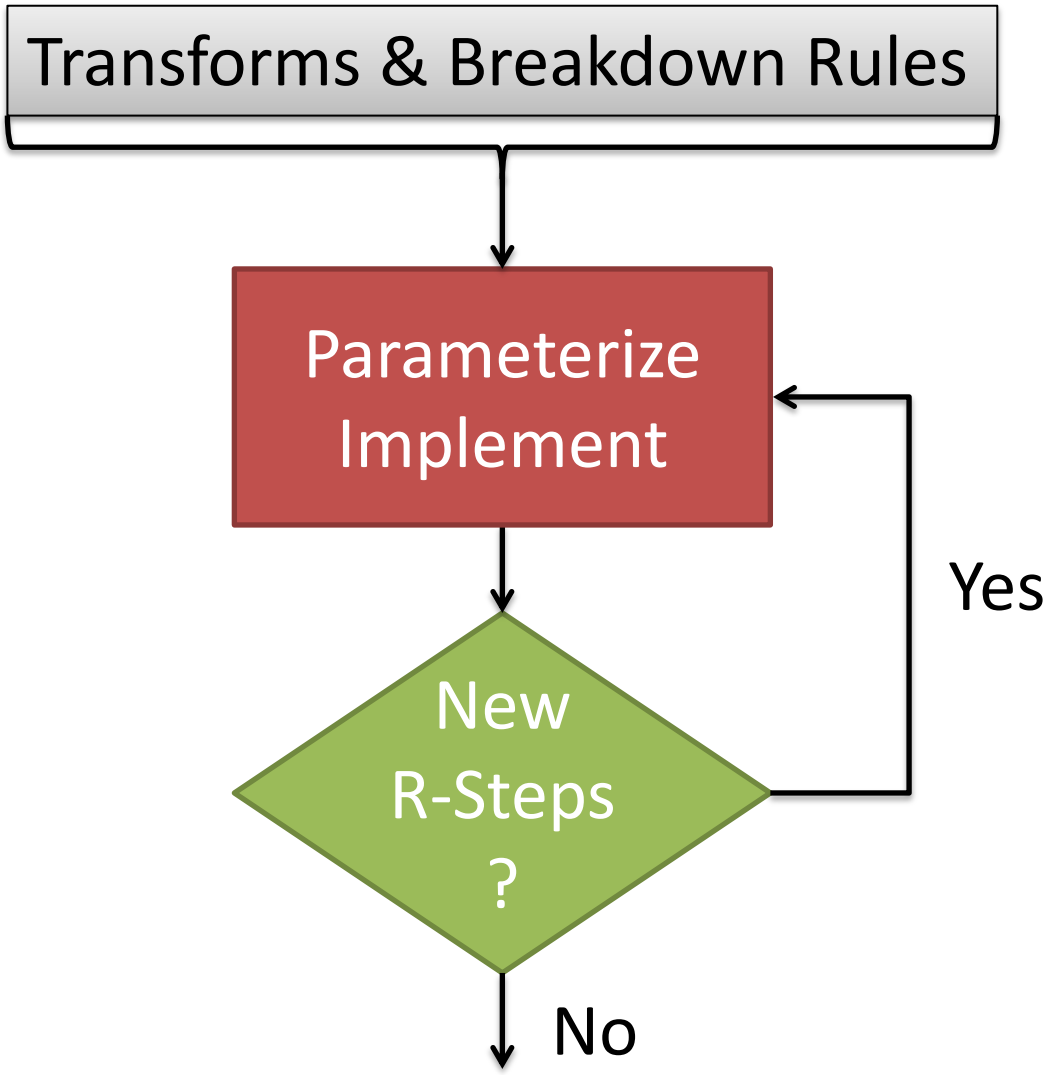
$$\mathbf{DFT_n} = (\mathbf{DFT_k} \otimes \mathrm{I_m})\ \mathrm{T^n_m}\ \ (\mathrm{I_k} \otimes \mathbf{DFT_m})\ \ \mathrm{L^n_k}$$

$$(I_k \otimes DFT_m)L^n_k = \left(\sum_{j=0}^{k-1} S\left(h_{jm,1}\right) DFT_m G(h_{jm,1})\right) \mathrm{perm}(\mathrm{l^m_k})$$

$$= \left(\sum_{j=0}^{k-1} S\left(h_{jm,1}\right) DFT_m G(h_{j,k})\right)$$

$$(DFT_k \otimes I_m)T^n_m = \left(\sum_{j=0}^{m-1} S\left(h_{j,m}\right) DFT_k diag(f \circ h_{i,k}) G(h_{j,m})\right)$$

Recursion Step Closure



Implement
expand & rewrite

Parametrize
find params
according to
constraints
(Matrix dimensions)

Recursion Steps - Example

$$\mathbf{DFT_n} = (\mathbf{DFT_k} \otimes \mathrm{I_m})\ \mathrm{T^n_m}\ (\mathrm{I_k} \otimes \mathbf{DFT_m})\ \mathrm{L^n_k}$$

`dft`

Function Tag to mark recursions

$$(I_k \otimes DFT_m)L^n_k = \left(\sum_{j=0}^{k-1} S(h_{jm,1})DFT_m G(h_{j,k})\right)$$

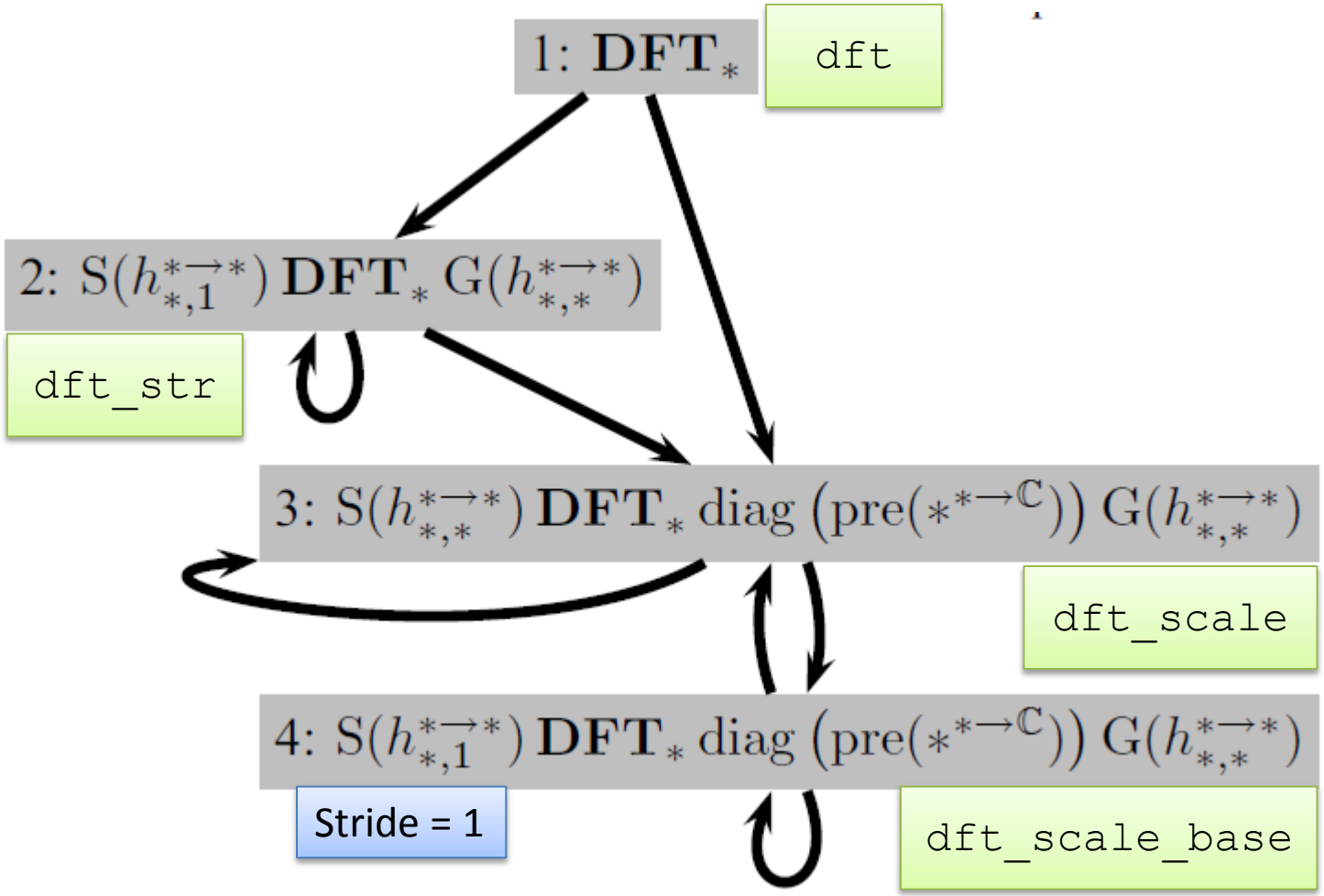
`dft_str`

$$(DFT_k \otimes I_m)T^n_m = \left(\sum_{j=0}^{m-1} S(h_{j,m})DFT_k diag(f \circ h_{i,k})G(h_{j,m})\right)$$

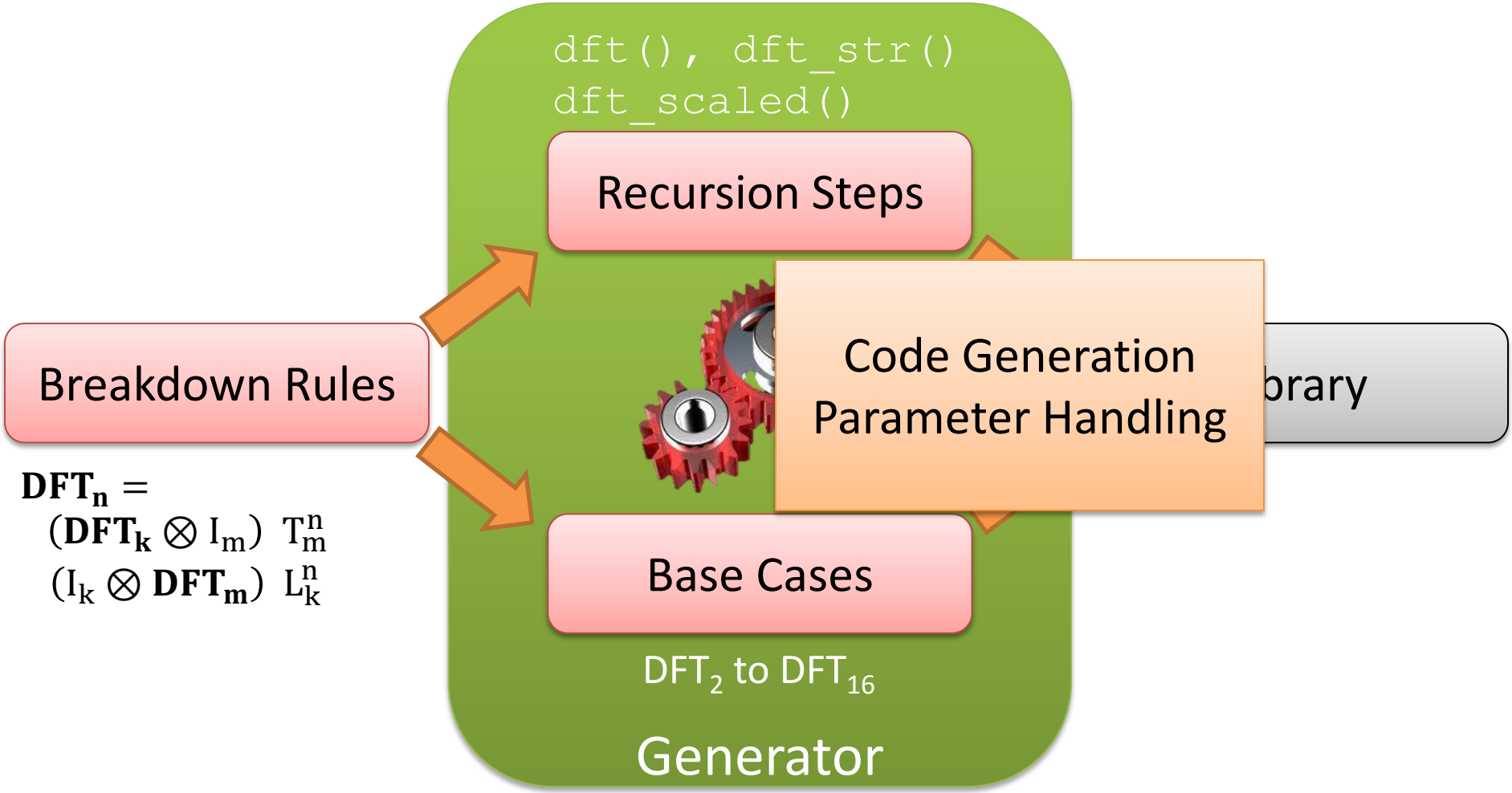
`dft_scaled`

Expand tag to scatter / gather

Recursion Steps – Graph

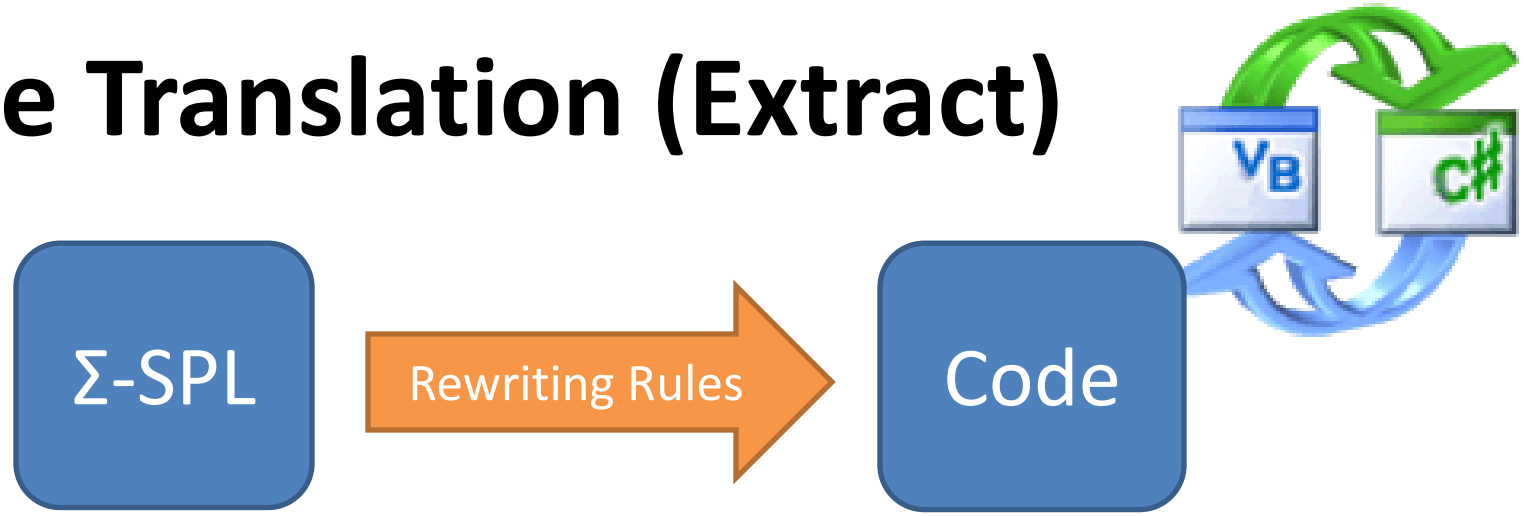


Library Generation for Transforms



http://www.anilgears.com/images1/gear6.jpg

Code Translation (Extract)



Code$(G(f^{n\to N}), y, x)$

```
for(j=0..n-1) y[j] = x[f(j)];
```

Code$(diag(f^{n\to\mathbb{C}}), y, x)$

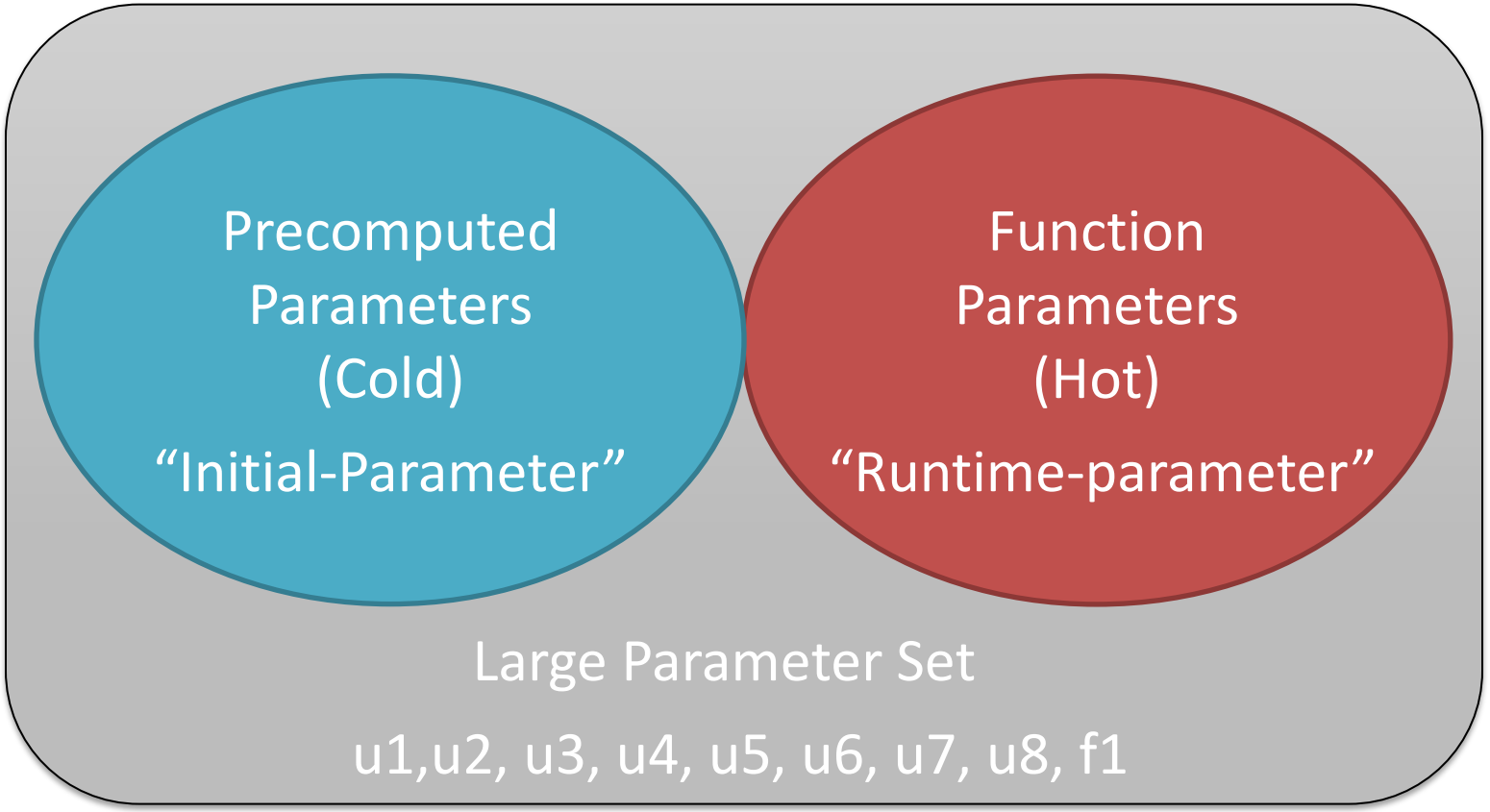
```
for(j=0..n-1) y[j] = f(j)*x[j];
```

Code$(\sum_{j=0}^{k-1} M_j, y, x)$

```
for(j=0..k-1)
```
Code(M_j, y, x)

http://www.carlosag.net/images/codeTranslatorLogo.gif

Parameter Handling



$$DFT_{u1} = S\left(h_{u3,1}^{u1\to u2}\right)\left(DFT_{f1} \otimes I_{u1/f1}\right) diag\left(pre\left(d_{u1/f1}^{u1\to\mathbb{C}}\right)\right) \cdot \left(I_{f1} \otimes DFT_{u1/f1}\right) L_{f1}^{u1} G\left(h_{u7,u8}^{u1\to u6}\right)$$

Library Generation

Generated Functions

Higher-order descriptor

```
Identified recursion steps
```

Cold parameters

```
DFT size n
```

Hot parameters

```
Vectors x,y
```

Base cases

```
DFT2
```

Actual Code

Classes

```
DFT, DFT_STR, DFT_SCALED
```

Constructor arguments

```
DFT(int n,…)
```

Method arguments

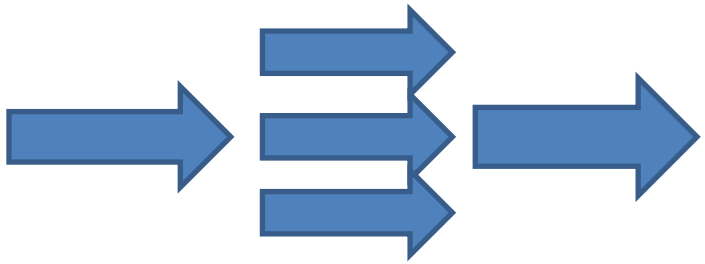
```
DFT->compute(x, y)
```

Constants

```
[1 1; 1 -1]
```

Library Generation: Platform

Parallel & vector code



New rewriting rules with

Processors, Cache-line size, vector length

Additional recursion steps:

DFT scalar:	4
DFT vectorized+parallel:	**8**

RESULTS

http://britishmastersaf.files.wordpress.com/2011/05/results.jpg

Test Plattform

CPU
2x dual core 3GHz Intel Xeon 5160 processors

Compiler:
Intel C/C++ Compiler 10.1 and SUN JDK 1.6.0

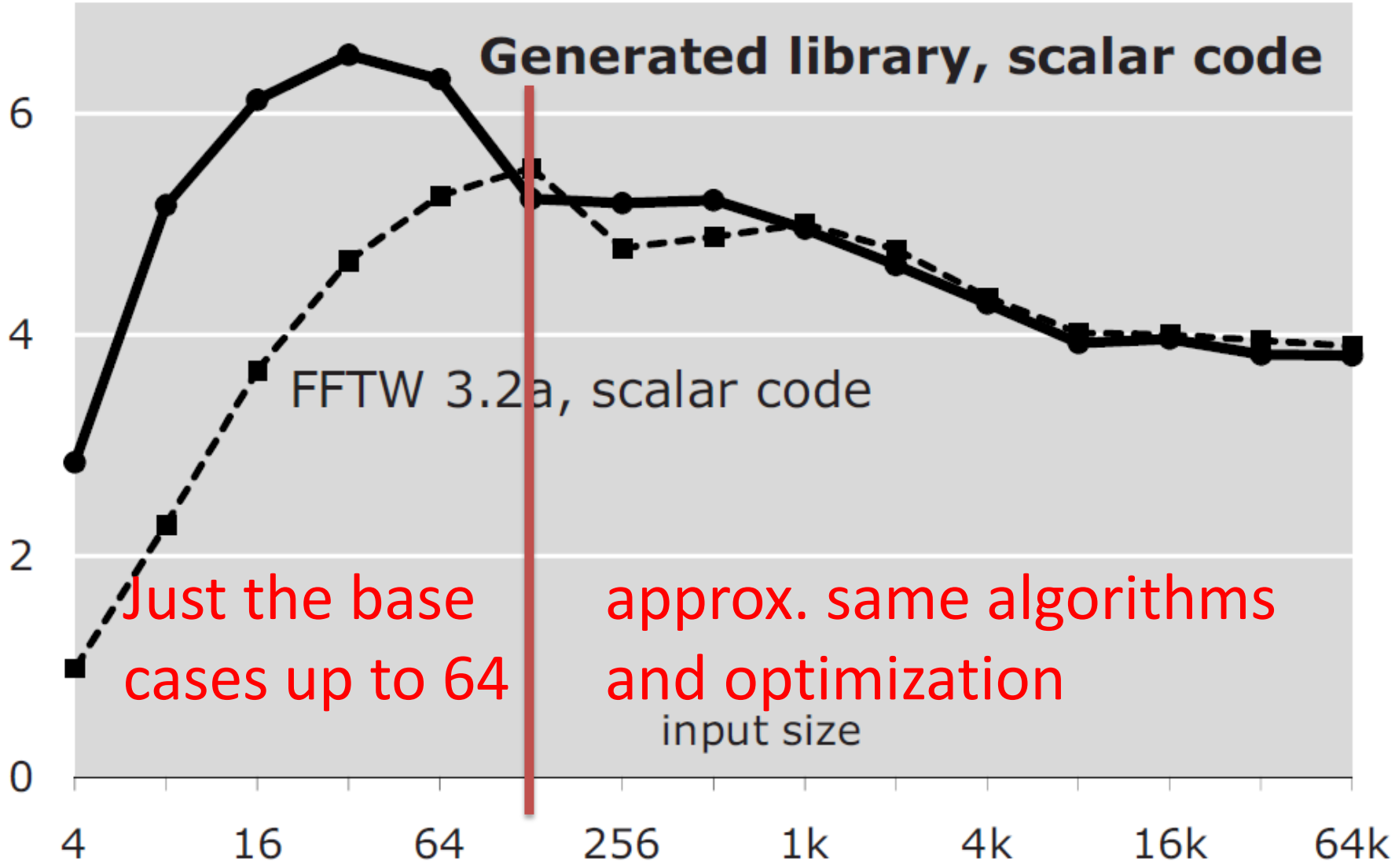
Threading:
OpenMP pragmas

Library Generation Time: between 10 and 60 minutes

Real DFT

Performance [Gflop/s]

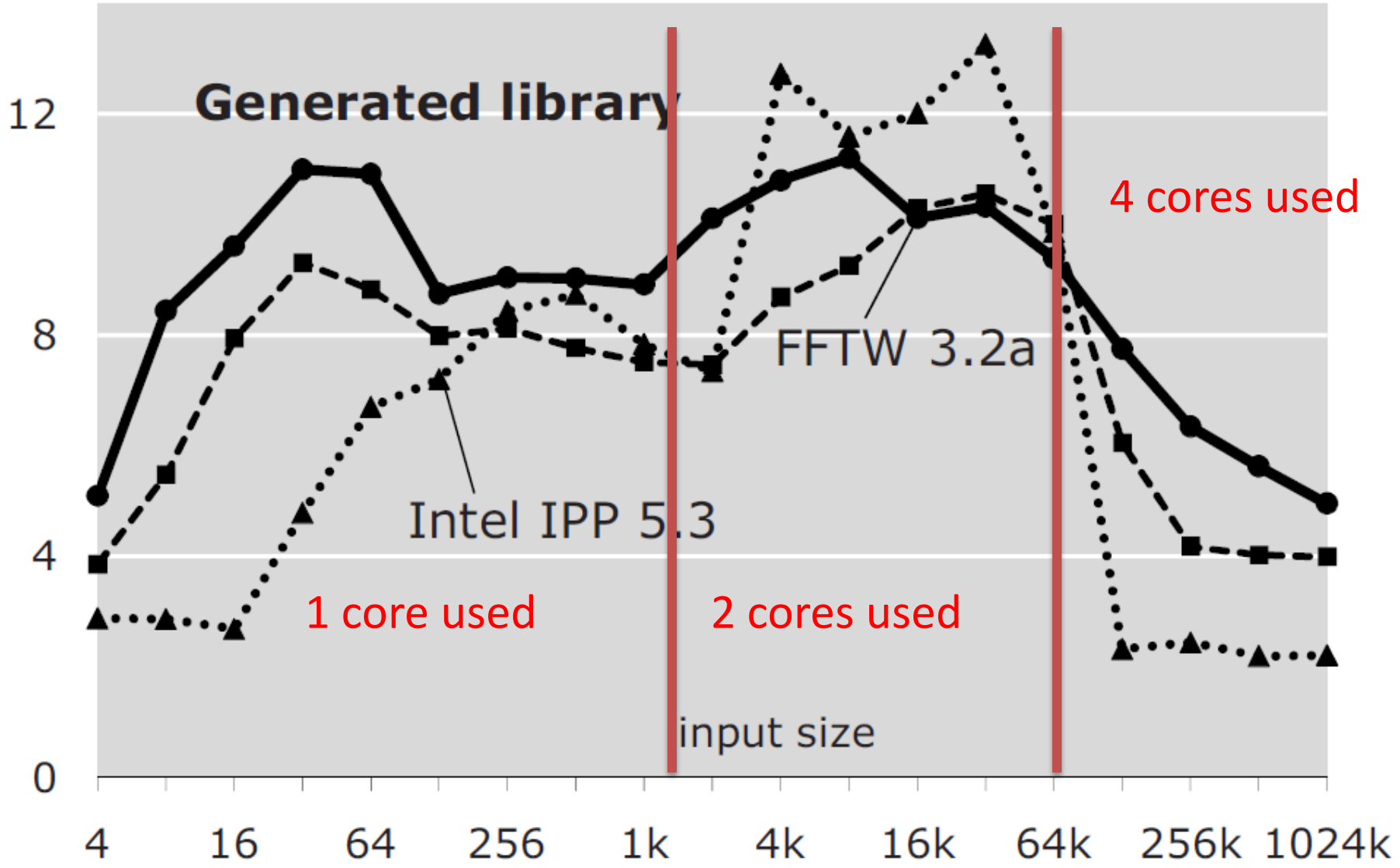
Scalar



Source: Paper

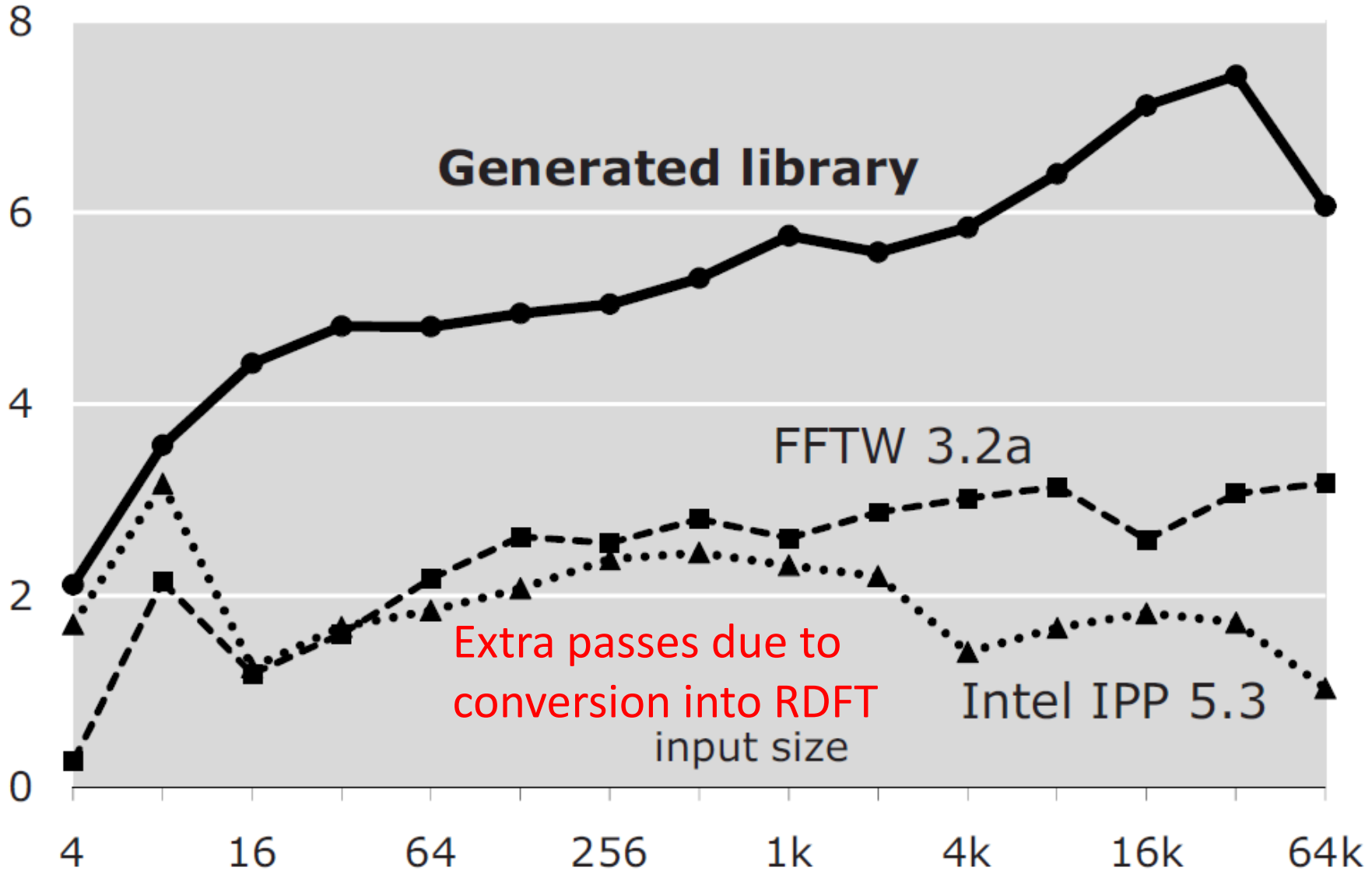
Complex DFT, double precision, up to 4 threads

Performance [Gflop/s]



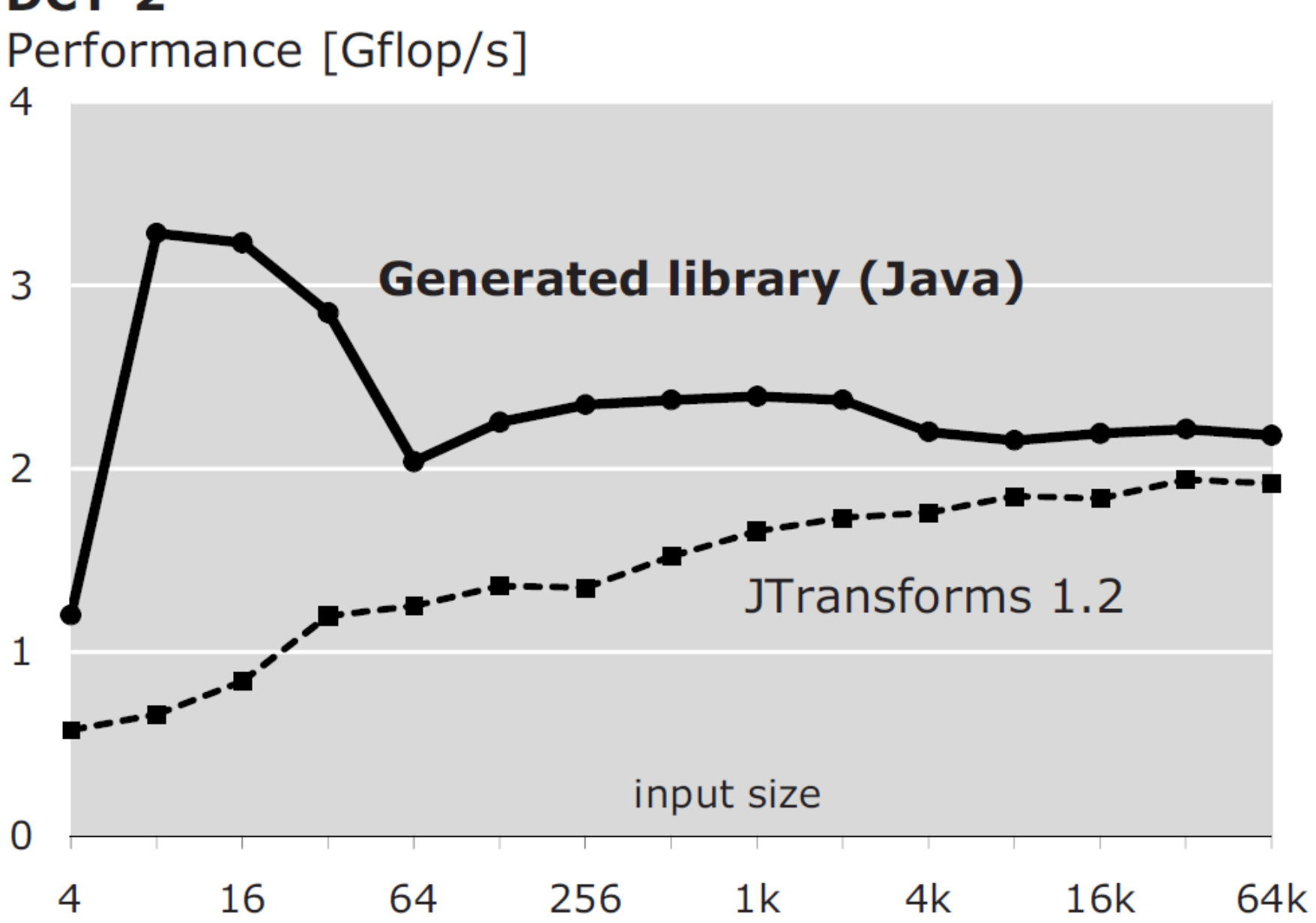
Source: Paper

DCT-2, double precision, up to 2 threads



Source: Paper

DCT-2



Source: Paper

Conclusion

+ Complete automation

+ Description at high level of abstraction

+ performance competitiveness with best human written code

+ flexibility: different custom libraries

Hot/Cold Partitioning