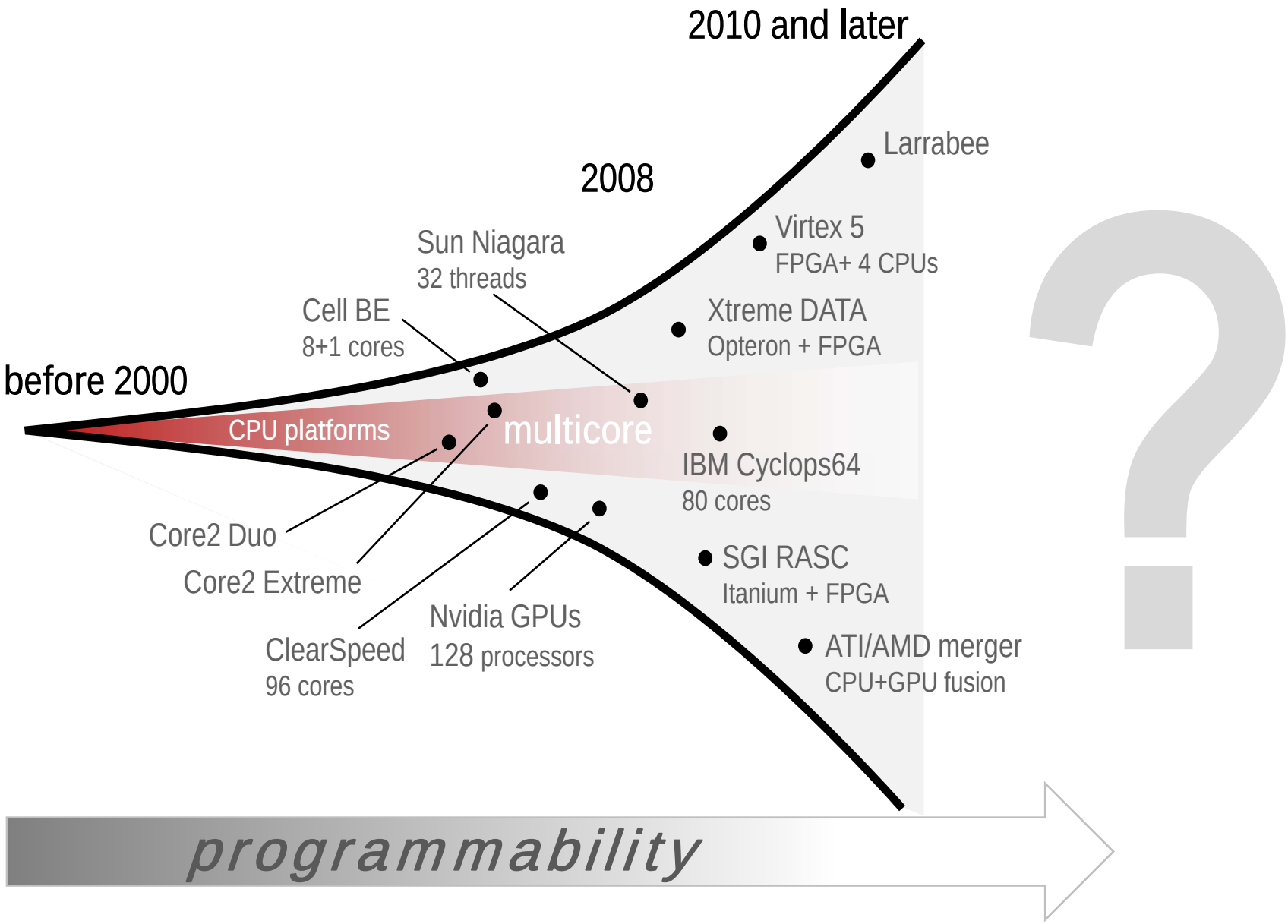
Meet Stevephanie, the Computer
Spiral: Automatic Library Generation

Franz Franchetti

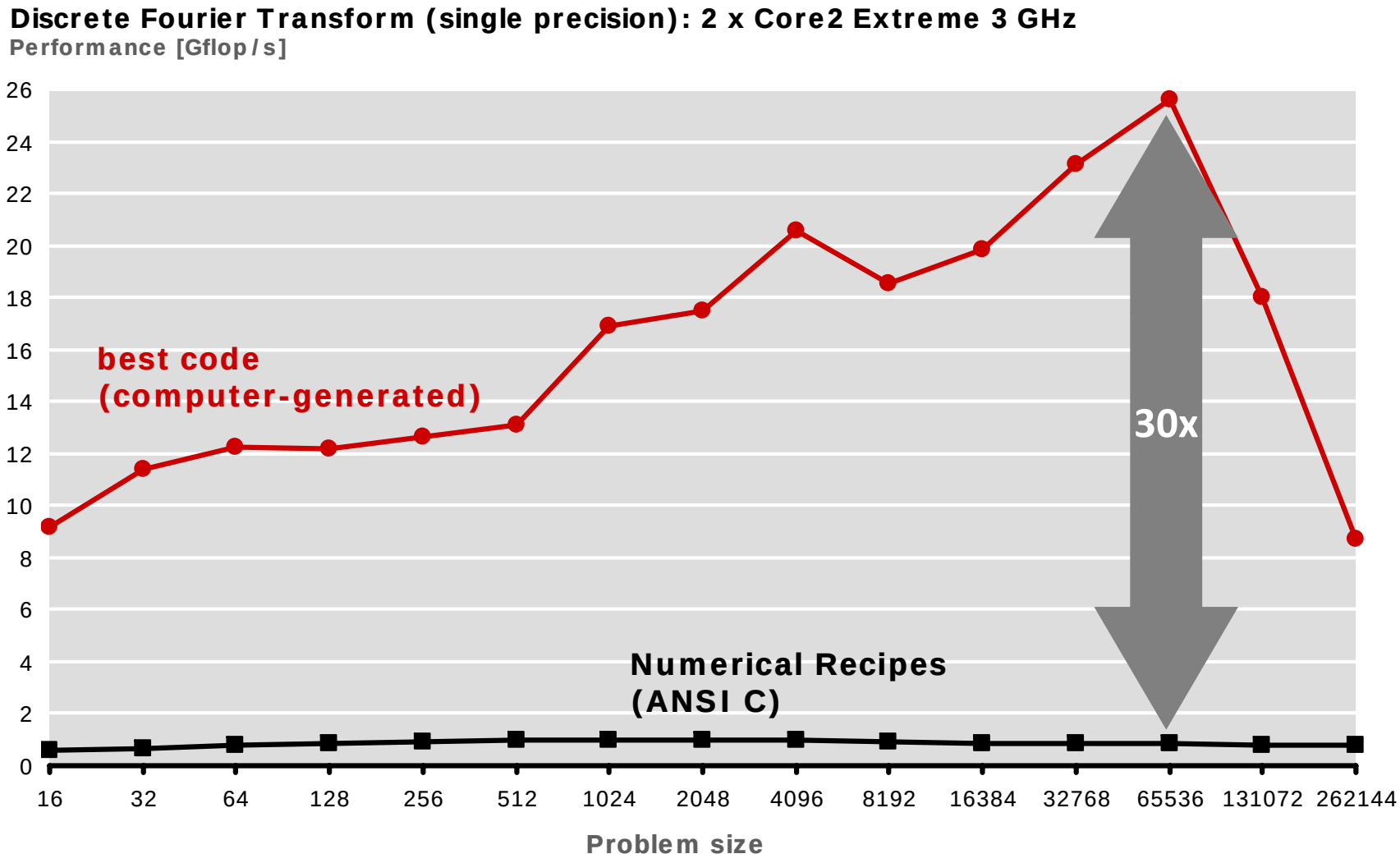
Electrical and Computer Engineering
Carnegie Mellon University

UW MSR Institute 2008, August 6th: "The Concurrency Challenge"

The Current Architecture Space

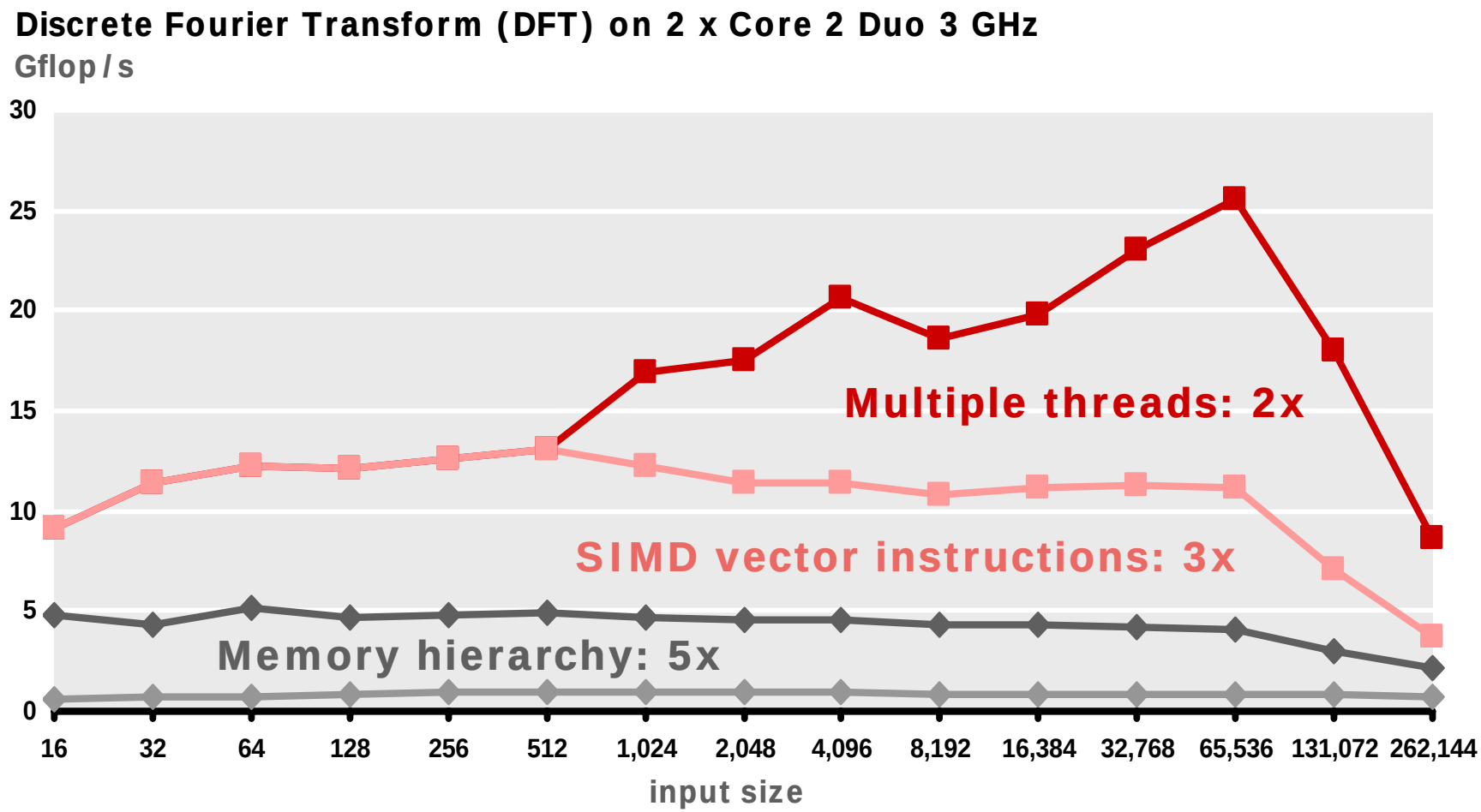


Example: DFT on "Nice" Quadcore



What's going on?

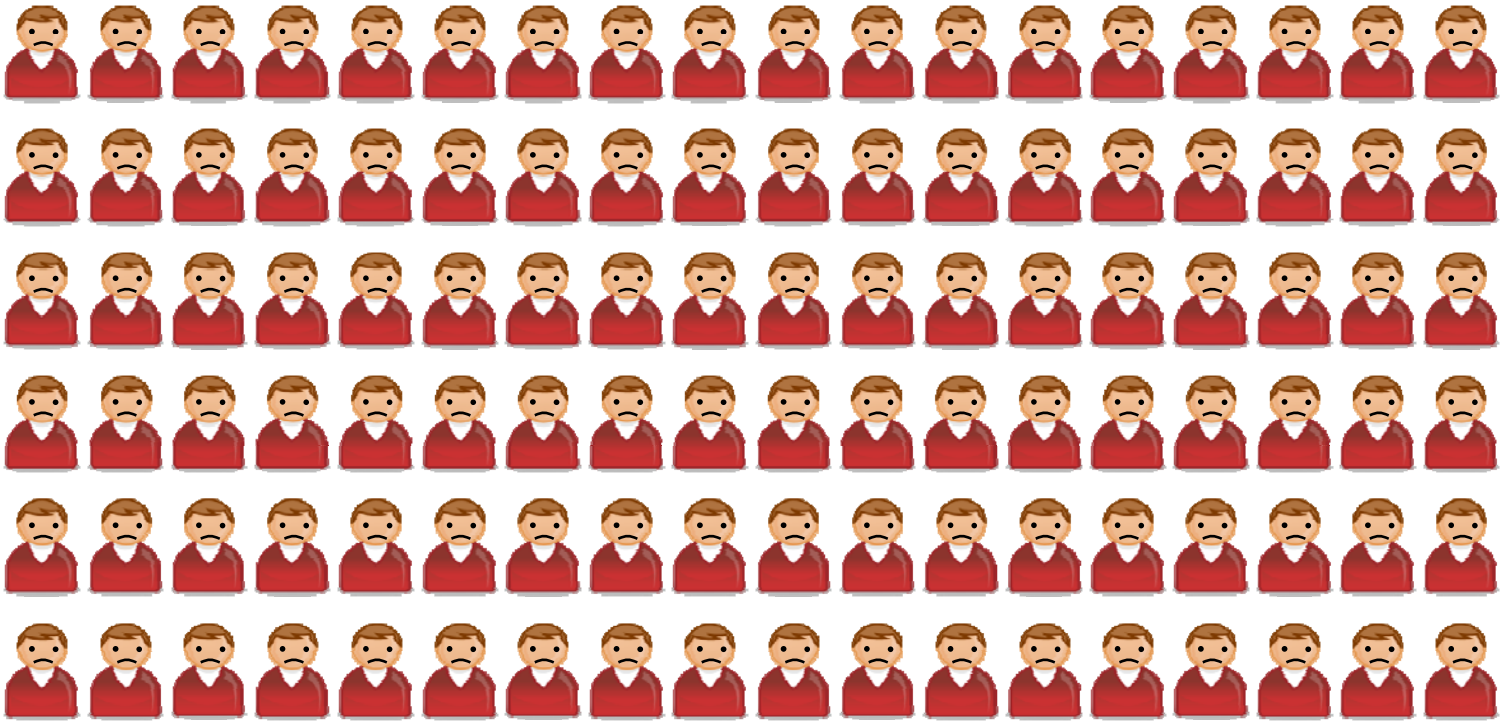
DFT Plot: Analysis



High performance is hard even on "simple" architectures

Current Solution

- **Legions of programmers implement and optimize the same functionality for every platform and whenever a new platform comes out.**

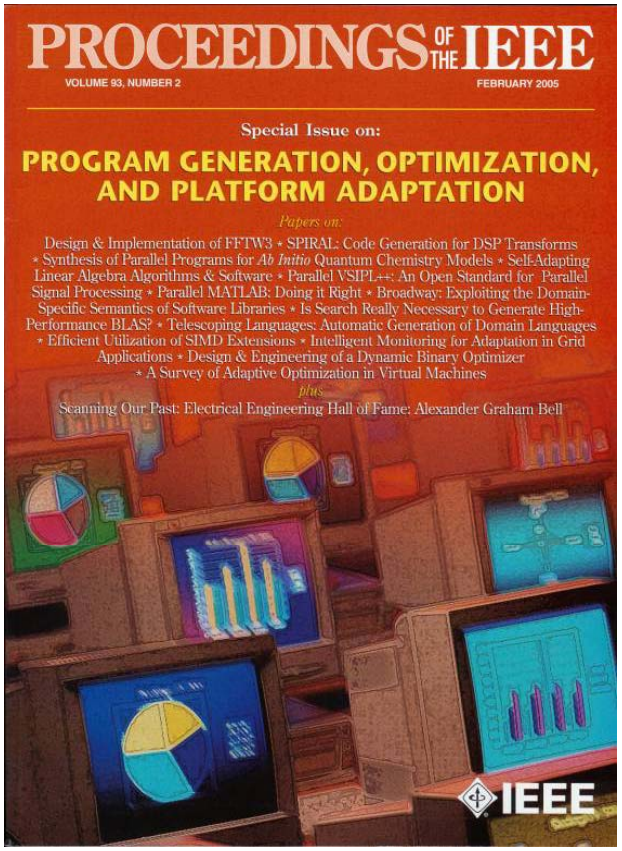


Better: Automatic Performance Tuning

- **Automate (parts of) the implementation or optimization**

- **Research efforts**
  - Linear algebra: Phipac/ATLAS, LAPACK, Sparsity/Bebop/OSKI, Flame
  - Tensor computations (TCE)
  - PDE/finite elements: Fenics
  - Adaptive sorting
  - Fourier transform: FFTW, UHFFT
  - Linear transforms: Spiral
  - …

Promising new area but more work needed
In particular for parallelism …



Proceedings of the IEEE special issue, Feb. 2005

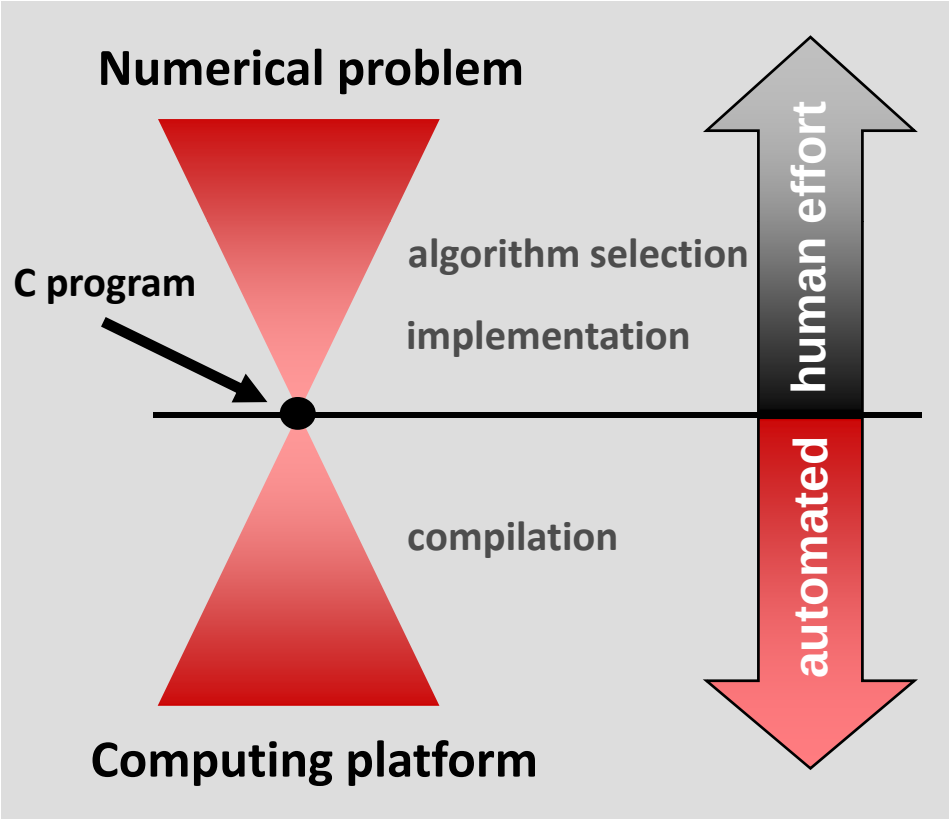
Spiral

- **Library generator for linear transforms (DFT, DCT, DWT, filters, ….)** ***and recently more …***
- **Wide range of platforms supported: scalar, fixed point, vector, parallel, Verilog, GPU**
- **Research Goal: "Teach" computers to write fast libraries**
  - Complete automation of implementation and optimization
  - Conquer the "high" algorithm level for automation
- **When a new platform comes out: Regenerate a retuned library**
- **When a new platform paradigm comes out (e.g., CPU+GPU): Update the tool rather than rewriting the library**

Intel is using Spiral to generate parts of their MKL and IPP libraries

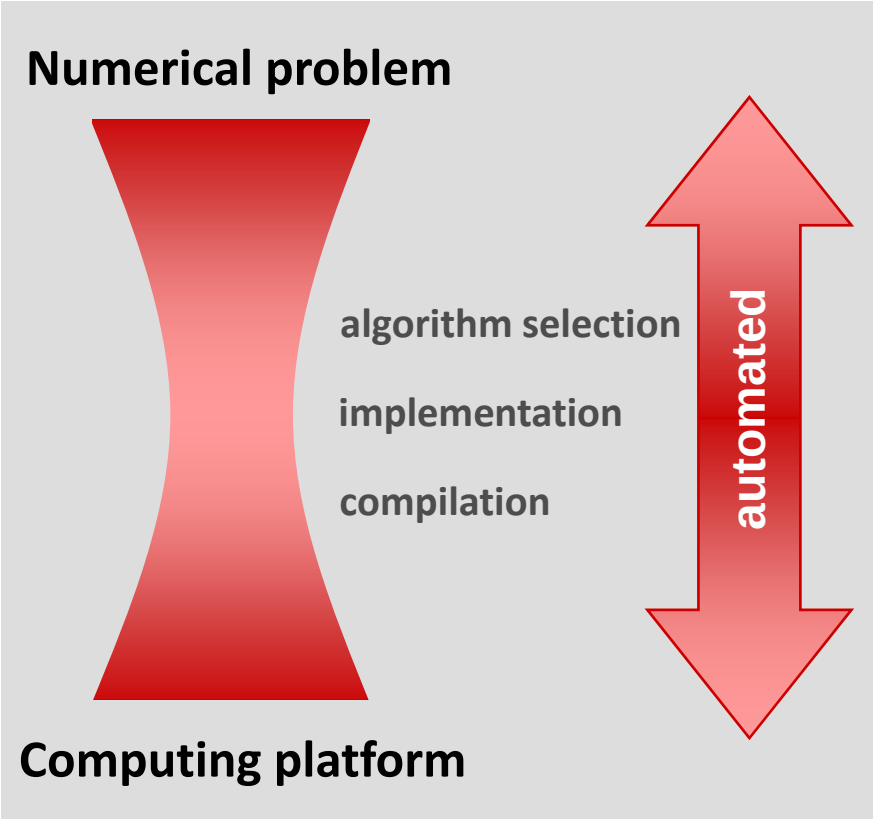
Vision Behind Spiral

Current



C code a singularity: Compiler has no access to high level information

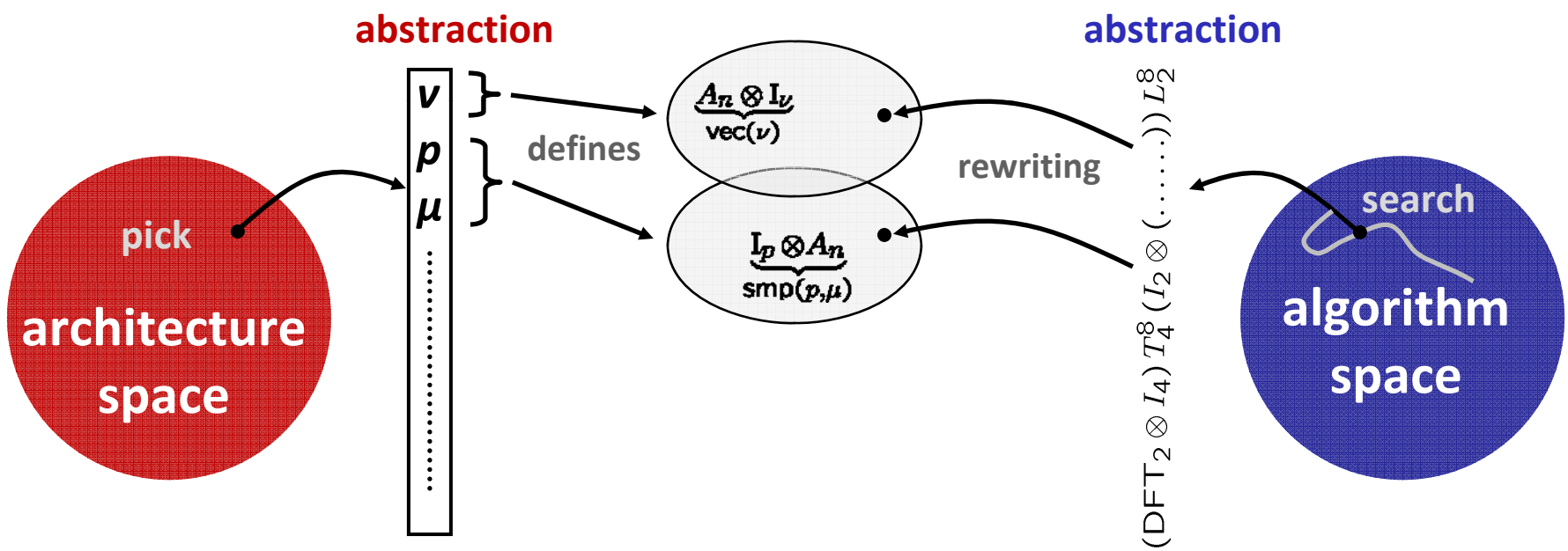
Future



Challenge: conquer the high abstraction level for complete automation

Main Idea: Joint Mathematical Abstraction

Model: common abstraction
= spaces of matching formulas



Architectural parameter:
Vector length,
#processors, …

optimization

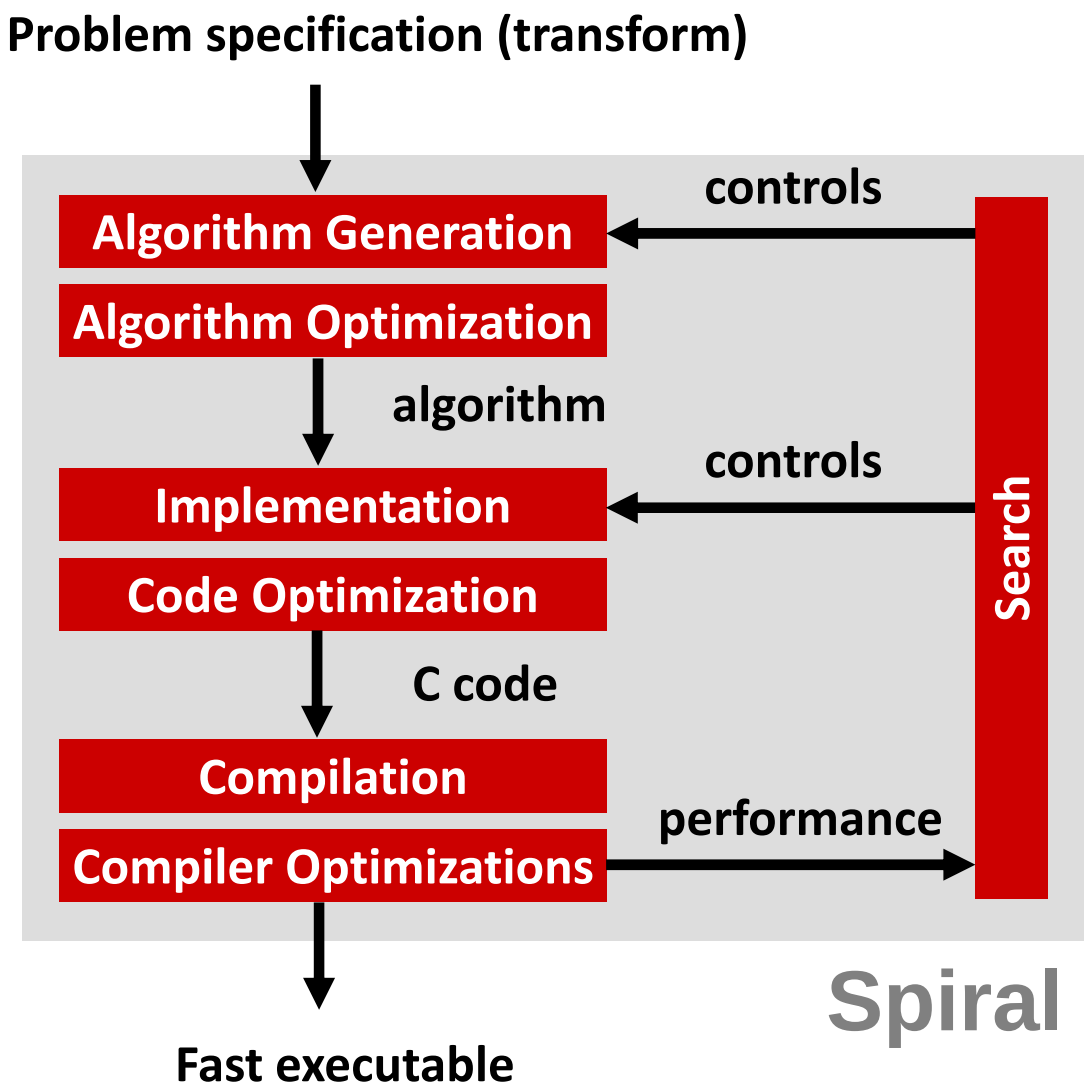
Kernel:
problem size,
algorithm choice

How Spiral Works

Spiral:
Complete automation of the implementation and optimization task

Basic idea:
Declarative representation of algorithms

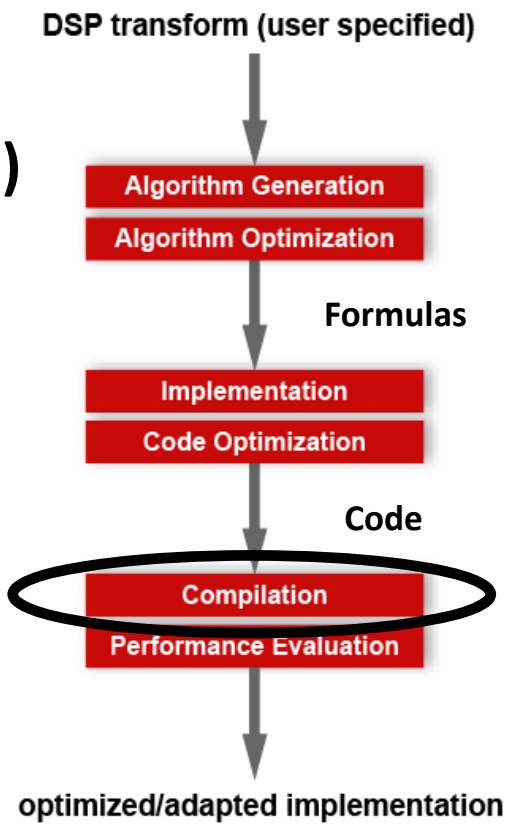
Rewriting systems to generate and optimize algorithms



Spiral

Generating, Not Finding Parallelism

- Shared Memory (SMP, CMP)
- Vector SIMD (SSE, VMX, Double FPU…)
- Message Passing (Clusters)
- DMA+Streaming (Cell BE)
- Graphics Processors (GPUs)
- Pipelining+Sreaming (FPGAs)
- HW/SW partitioning (CPU+FPGA)



One methodology optimizes for all types of parallelism

Program Generation in Spiral (Sketched)

Transform
user specified

$$\mathrm{DFT}_8$$

Fast algorithm in SPL
many choices

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

∑-SPL:

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

C Code:

```
void sub(double *y, double *x) {
  double f0, f1, f2, f3, f4, f7, f8, f10, f11;
  f0 = x[0] - x[3];
  f1 = x[0] + x[3];
  f2 = x[1] - x[2];
  f3 = x[1] + x[2];
  f4 = f1 - f3;
  y[0] = f1 + f3;
  y[2] = 0.7071067811865476 * f4;
  f7 = 0.9238795325112867 * f0;
  f8 = 0.3826834323650898 * f2;
  y[1] = f7 + f8;
  f10 = 0.3826834323650898 * f0;
  f11 = (-0.9238795325112867) * f2;
  y[3] = f10 + f11;
}
```

Optimization at all abstraction levels

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

Behind The Scenes: Domain-Specific Rewriting

Algorithms

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

$$\mathbf{DFT}_n \rightarrow P_n(\mathbf{DFT}_k \otimes \mathbf{DFT}_m)Q_n, \quad n = km,\ \gcd(k,m) = 1$$

$$\mathbf{DFT}_p \rightarrow R_p^T(\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1})D_p(\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1})R_p, \quad p \text{ prime}$$

DCT-3

DCT-4

IMDCT$_{2}$

DCT-4$_{2m}$

- "Teaches" Spiral about existing algorithm knowledge (~200 journal papers)
- "Teaches" Spiral about hardware properties
- "Teaches" Spiral about program transformations

Program transformations

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

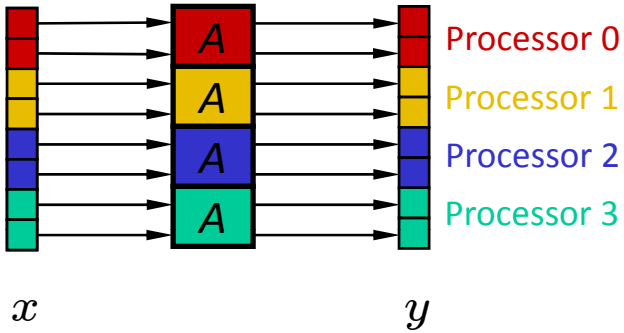
Hardware properties

$$\underbrace{\mathsf{L}^{mn}_m}_{\mathsf{smp}(p,\mu)} \rightarrow \begin{cases} \underbrace{\left(\mathrm{I}_p \otimes \mathsf{L}^{mn/p}_{m/p}\right)}_{\mathsf{smp}(p,\mu)} \underbrace{\left(\mathsf{L}^{pn}_p \otimes \mathrm{I}_{m/p}\right)}_{\mathsf{smp}(p,\mu)} \\ \underbrace{\left(\mathsf{L}^{pm}_m \otimes \mathrm{I}_{n/p}\right)}_{\mathsf{smp}(p,\mu)} \underbrace{\left(\mathrm{I}_p \otimes \mathsf{L}^{mn/p}_m\right)}_{\mathsf{smp}(p,\mu)} \end{cases}$$

Formulas Encode Program Properties

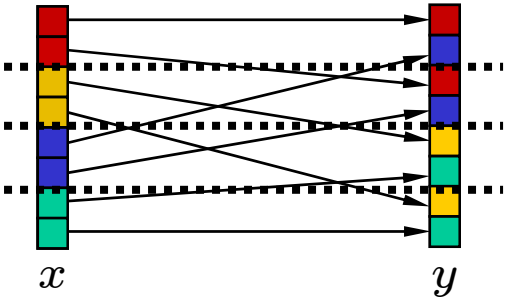
- **Embarrassingly parallel, load balanced computation**

$$y = \left(\mathrm{I}_p \otimes A \right) x$$



- **False sharing**

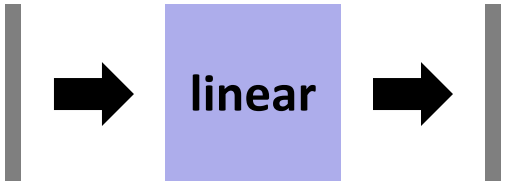
$$y = \mathrm{L}^8_4 \, x$$



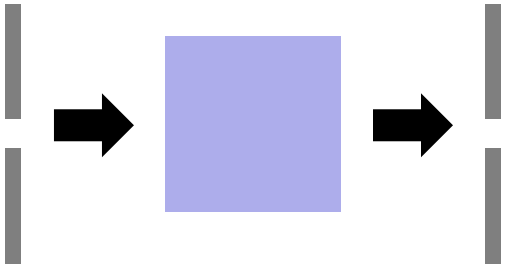
Span space of false-sharing free programs for empirical tuning

Going Beyond Transforms

- **Transform =**
  linear operator with one vector input and one vector output



- **Key ideas:**
  - Generalize to (possibly nonlinear) operators with several inputs and several outputs
  - Generalize SPL (including tensor product) to OL (operator language)
  - Generalize rewriting systems for parallelizations



Expressing Kernels as Operator Formulas

Matrix-Matrix Multiplication

$$\mathsf{MMM}_{1,1,1} \rightarrow (\cdot)_1$$

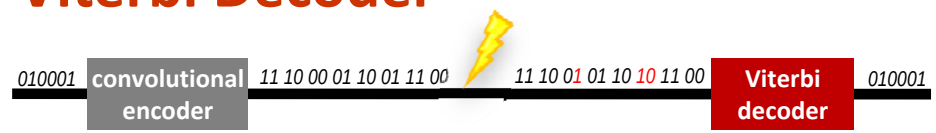
$$\mathsf{MMM}_{m,n,k} \rightarrow (\otimes)_{m/m_b \times 1} \otimes \mathsf{MMM}_{m_b,n,k}$$

$$\mathsf{MMM}_{m,n,k} \rightarrow \mathsf{MMM}_{m,nb,k} \otimes (\otimes)_{1 \times n/nb}$$

$$\mathsf{MMM}_{m,n,k} \rightarrow ((\Sigma_{k/k_b} \circ (\cdot)_{k/k_b}) \otimes \mathsf{MMM}_{m,n,k_b}) \circ ((L^{mk/k_b}_{k/k_b} \otimes I_{k_b}) \times I_{kn})$$

$$\mathsf{MMM}_{m,n,k} \rightarrow (L^{mn/n_b}_{m} \otimes I_{n_b}) \circ ((\otimes)_{1 \times n/n_b} \otimes \mathsf{MMM}_{m,n_b,k}) \circ (I_{km} \times (L^{kn/n_b}_{n/n_b} \otimes I_{n_b}))$$

Viterbi Decoder



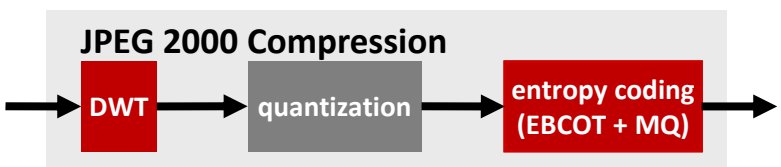
$$\underbrace{\mathrm{Vit}}_{\mathrm{vec}(v)} \rightarrow \underbrace{\left(\prod (L \times I) \circ (I \otimes C)\right) \circ Id}_{\mathrm{vec}(v)}$$

$$\rightarrow \left(\prod \underbrace{(L \times I) \circ (I \otimes C)}_{\mathrm{vec}(v)}\right) \circ Id$$

$$\times \quad \rightarrow \left(\prod (L \otimes I_v \times I) \circ (I \otimes C \otimes I_v) \circ (\vec{L} \times I)\right) \circ Id$$

$$\rightarrow \prod (L \otimes I_v \times I) \circ (I \otimes (B \otimes I_v)) \circ (\vec{L} \times I)$$

JPEG 2000 (Wavelet, EBCOT)



$$\mathrm{SC}(\chi_{m,n}, \sigma_{m,n}) : (\mathbb{Z}_2^9 \times \mathbb{Z}_2^9) \rightarrow (\mathbb{N}, \mathbb{Z}_2)$$

$$(I \times \mathrm{xor}_2) \circ (T_{SC} \times I) \circ (H \times V \times I) \circ (\underline{L_4^2} \times G_4) \circ (\underline{\begin{pmatrix} 1 \\ 1 \end{pmatrix}} \times \underline{\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}})$$

$$\mathrm{H}, \mathrm{V} : (\mathbb{Z}_2^9 \times \mathbb{Z}_2^9) \rightarrow \mathbb{N}$$

$$\mathrm{H}: \quad h \circ (f \times f) \circ (G_1 \times \mathrm{C}_{-2} \times G_1 \times G_7 \times \mathrm{C}_{-2} \times G_7) \circ \underline{L_4^2} \circ (\underline{\binom{1}{1}} \times \underline{\binom{1}{1}})$$

$$\mathrm{V}: \quad h \circ (f \times f) \circ (G_3 \times \mathrm{C}_{-2} \times G_3 \times G_5 \times \mathrm{C}_{-2} \times G_5) \circ \underline{L_4^2} \circ (\underline{\binom{1}{1}} \times \underline{\binom{1}{1}})$$

$$f : \mathrm{mul}_2 \circ (I \times \mathrm{sub}_2) \circ (I \times \mathrm{C}_1 \times \mathrm{mul}_2)$$

$$h : \mathrm{min}_2 \circ (\mathrm{C}_1 \times \mathrm{max}_2) \circ (\mathrm{C}_{-1} \times \mathrm{sum}_2)$$

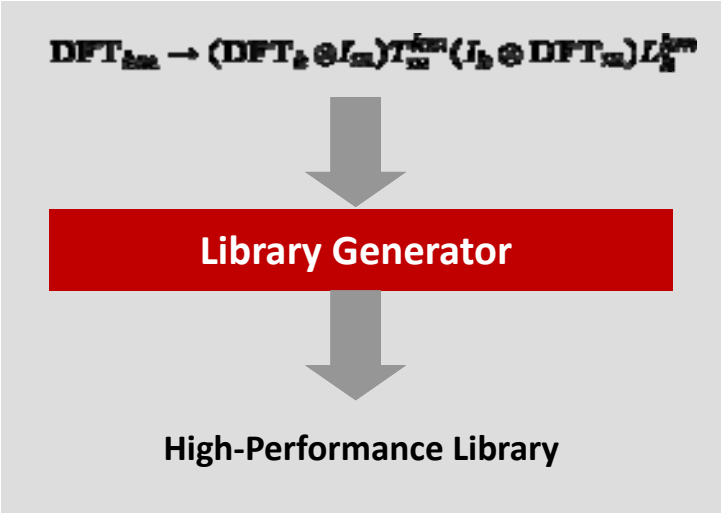
Synthetic Aperture Radar (SAR)

$$\begin{aligned}
\mathrm{SAR} &\rightarrow \text{2D-iDFT} \circ \mathrm{Interpl} \circ \mathrm{MatchFilt} \circ \mathrm{prep} \\
\text{2D-iDFT} &\rightarrow \mathrm{iDFT} \otimes \mathrm{iDFT} \\
\mathrm{MatchFilt} &\rightarrow \mathrm{Filt} \circ (\mathrm{I} \times \mathrm{C}_f) \\
\mathrm{Filt} &\rightarrow (\mathrm{I} \otimes (\cdot)) \\
\mathrm{Interpl} &\rightarrow (\Sigma \otimes \mathrm{I}) \circ (\mathrm{I} \otimes_j \mathrm{S}_{x_j \otimes y_j}) \circ \mathrm{Filt} \circ ((\mathrm{I} \otimes \mathbf{1} \otimes \mathrm{I}) \times \mathrm{I}) \circ (\mathrm{I} \times \mathrm{C}_{\iota \otimes_j g_j})
\end{aligned}$$

General-Size Library

Input:

- **Transform**: $\mathbf{DFT}_n$
- **Algorithms**: $\mathbf{DFT}_{km} \rightarrow (\mathbf{DFT}_k \otimes I_m) T^{km}_m (I_k \otimes \mathbf{DFT}_m) L^{km}_k$
  $\mathbf{DFT}_2 \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- **Vectorization**: 2-way SSE
- **Threading**: Yes



Output:

- Optimized library (10,000 lines of C++)
- For general input size
  (**not** collection of fixed sizes)
- Vectorized
- Multithreaded
- With runtime adaptation mechanism
- Performance competitive with hand-written code

Generating General-Size Libraries

$$\mathbf{DFT}_{km} \rightarrow (\mathbf{DFT}_k \otimes I_m) T^{km}_m (I_k \otimes \mathbf{DFT}_m) L^{km}_k$$

Spiral Library Generator

Recursion

```
void compute(Y, X, …) {
  for(i=0..k-1) c1->compute(…)
  for(j=0..m-1) c2->compute(…)
}
```

Base cases

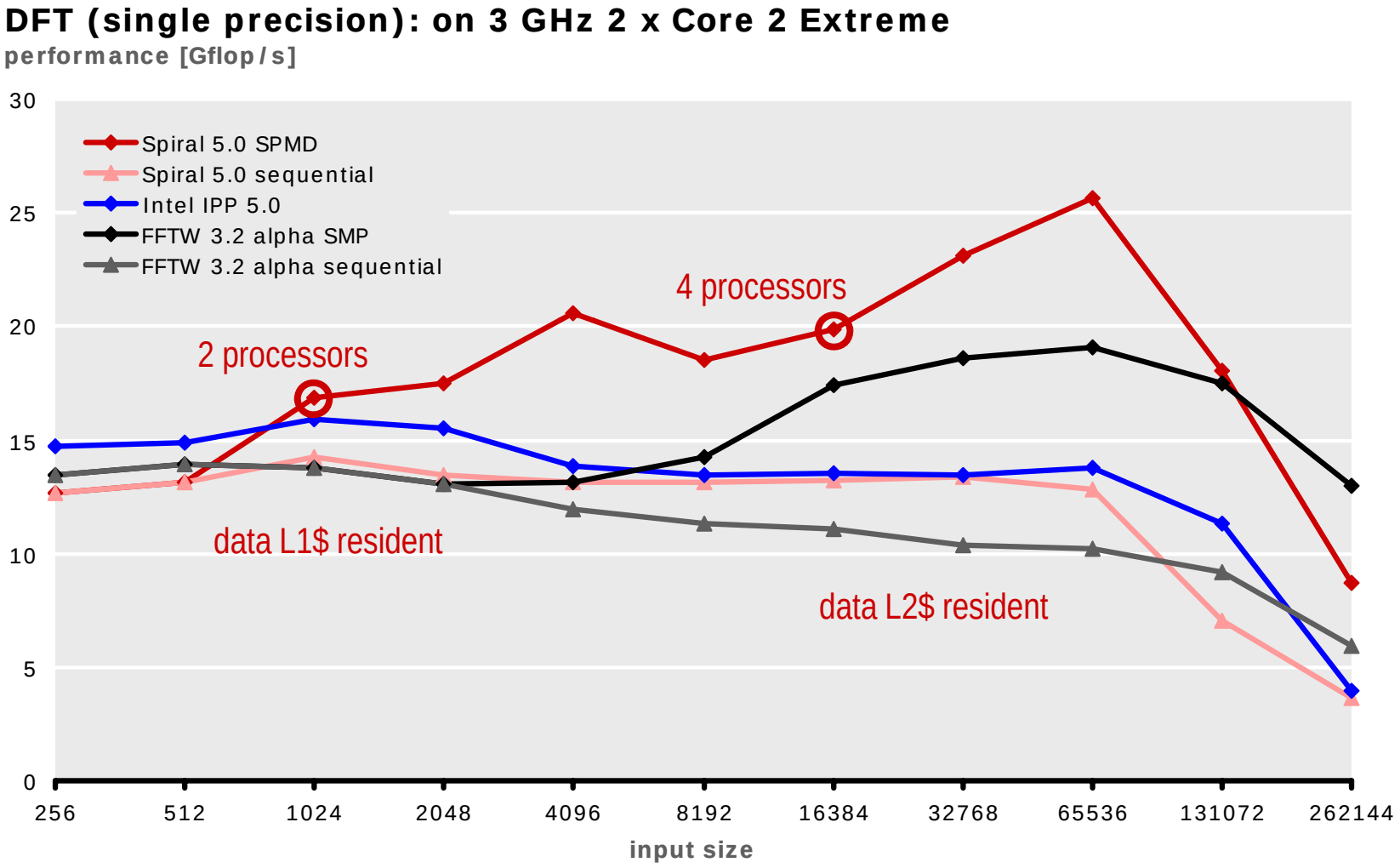
```
void dft2 (…) {
   y[0]=x[0]+x[1]
   y[1]=x[0]-x[1]
}
```

Target Infrastructure

```
fftw_plan_dft_1d(…) {
   hand->buf=
       fftw_malloc(…);
   …
}
```

Spiral generates whole library: FFTW equivalent and more

Benchmark: DFT on "Nice" Quadcore



4-way vectorized + up to 4-threaded + adapted to the memory hierarchy

Summary

- **Libraries must be well-tuned**
  Nightmare: locality, multicore, special instructions,…
- **Automating the tuning process is desirable**
  Computer-generated code can beat hand-tuned code
- **High-level abstraction is key to performance portability**
  That does not imply "easy to program"
- **Research challenge: Automate more domains**
  Program generators must become part of the ecosystem

Acknowledgement: Spiral

www.spiral.net

Joint work with
Yevgen Voronenko
Frédéric de Mesmay
Markus Püschel

... and the Spiral team (only part shown)

This work was supported by
DARPA DESA program, NSF-NGS/ITR, NSF-ACR, Mercury Inc., and Intel