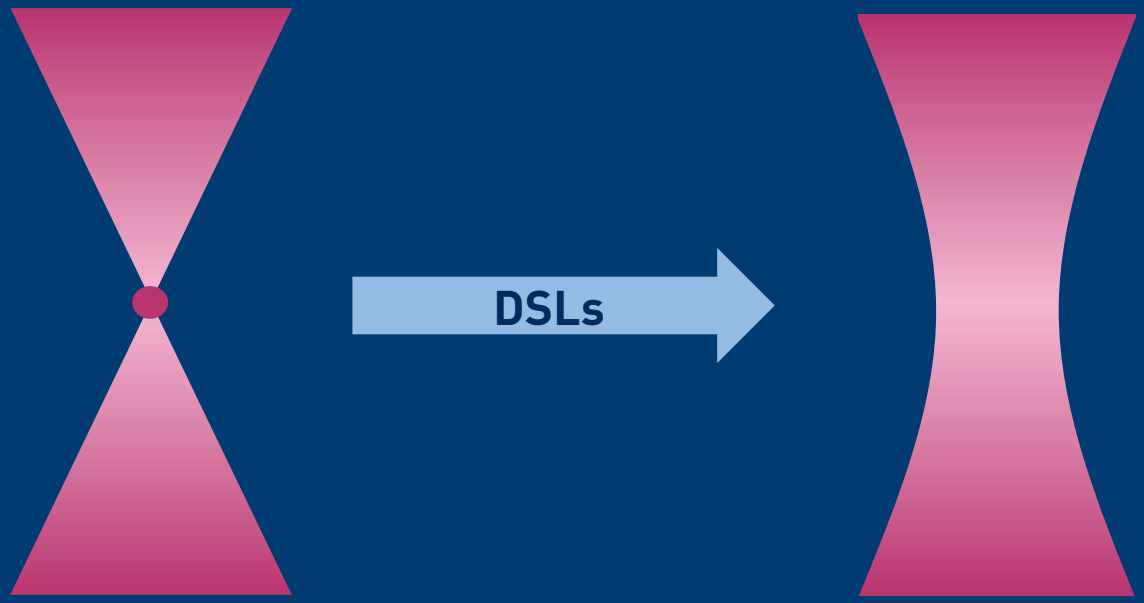


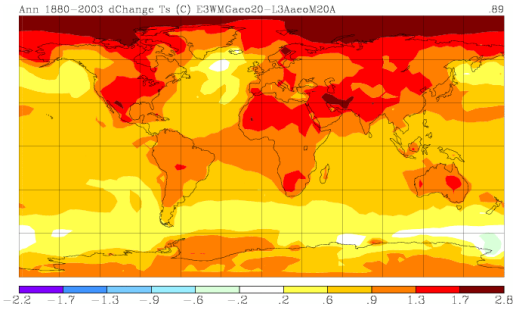
High Performance: How DSLs Can Help

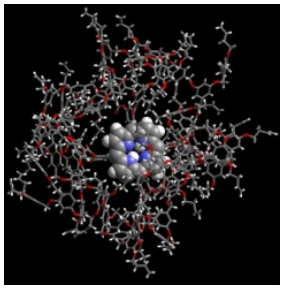
Markus Püschel

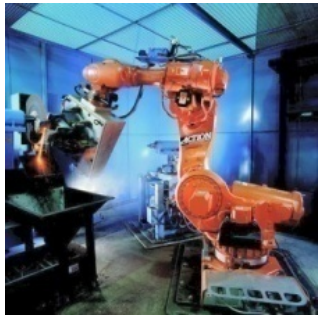
Computer Science
ETH zürich

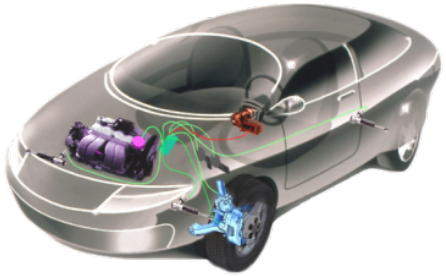
SPIRAL
www.spiral.net

Computing









Science simulations

Audio, image, Video processing

Signal processing, communication, control

Security

Machine learning, data analytics

Optimization

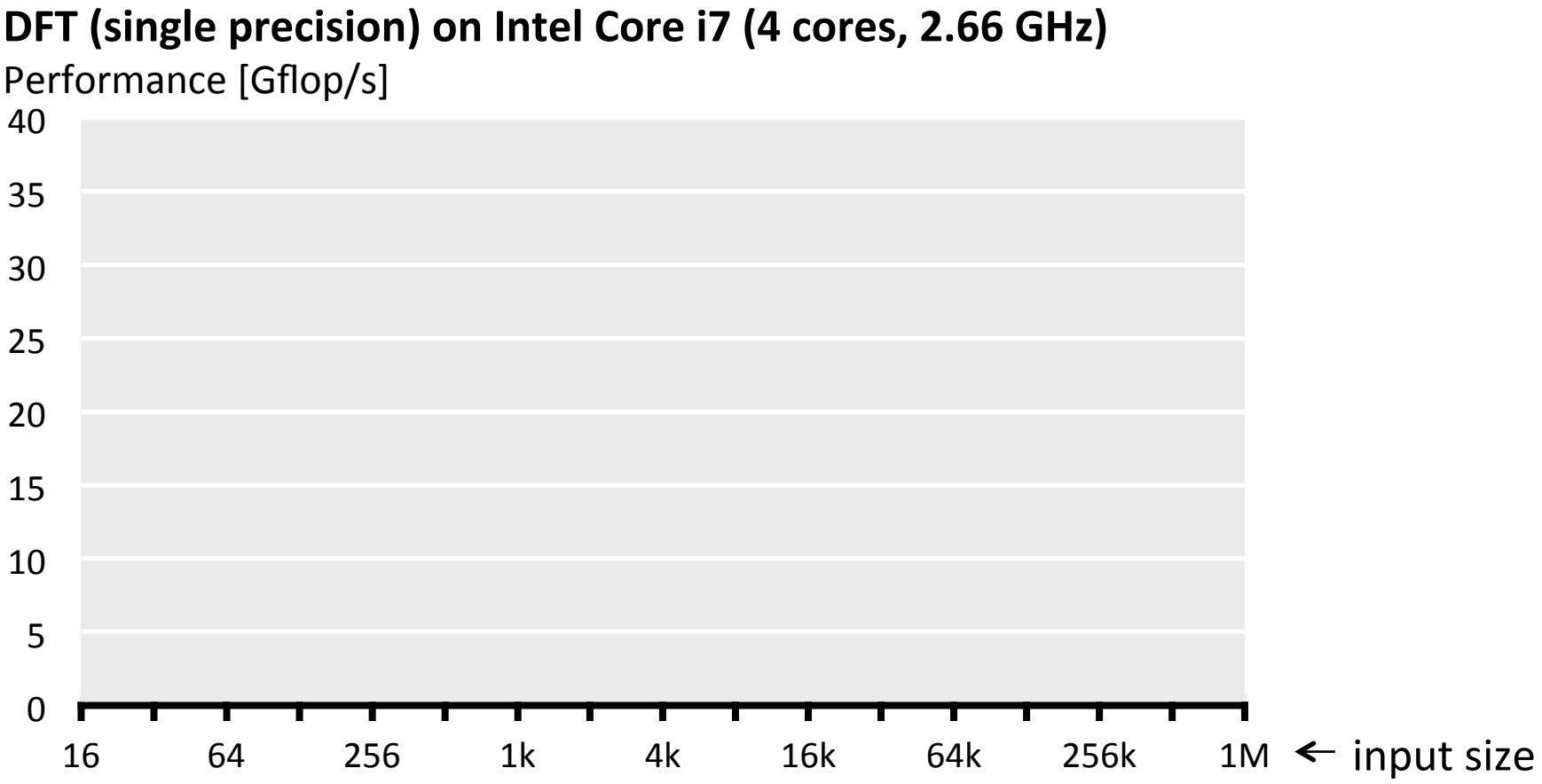
Highest performance is often crucial

How Do We Get Fast Code?

Algorithms	Choose cheap algorithm
Software	Implement in C/C++
Compilers	Choose good compiler and flags
Microarchitecture	Runs very fast

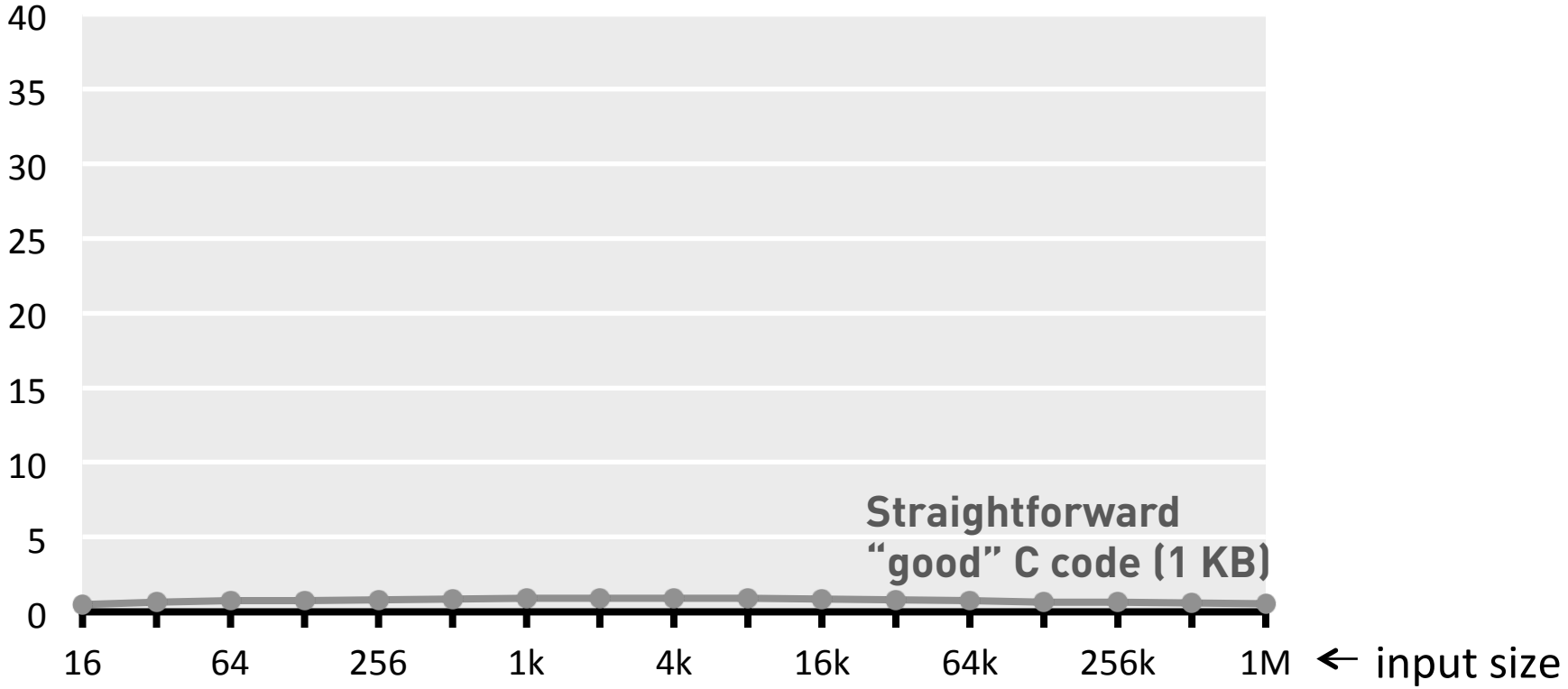
How well does this work?

Example: Discrete Fourier Transform



Example: Discrete Fourier Transform

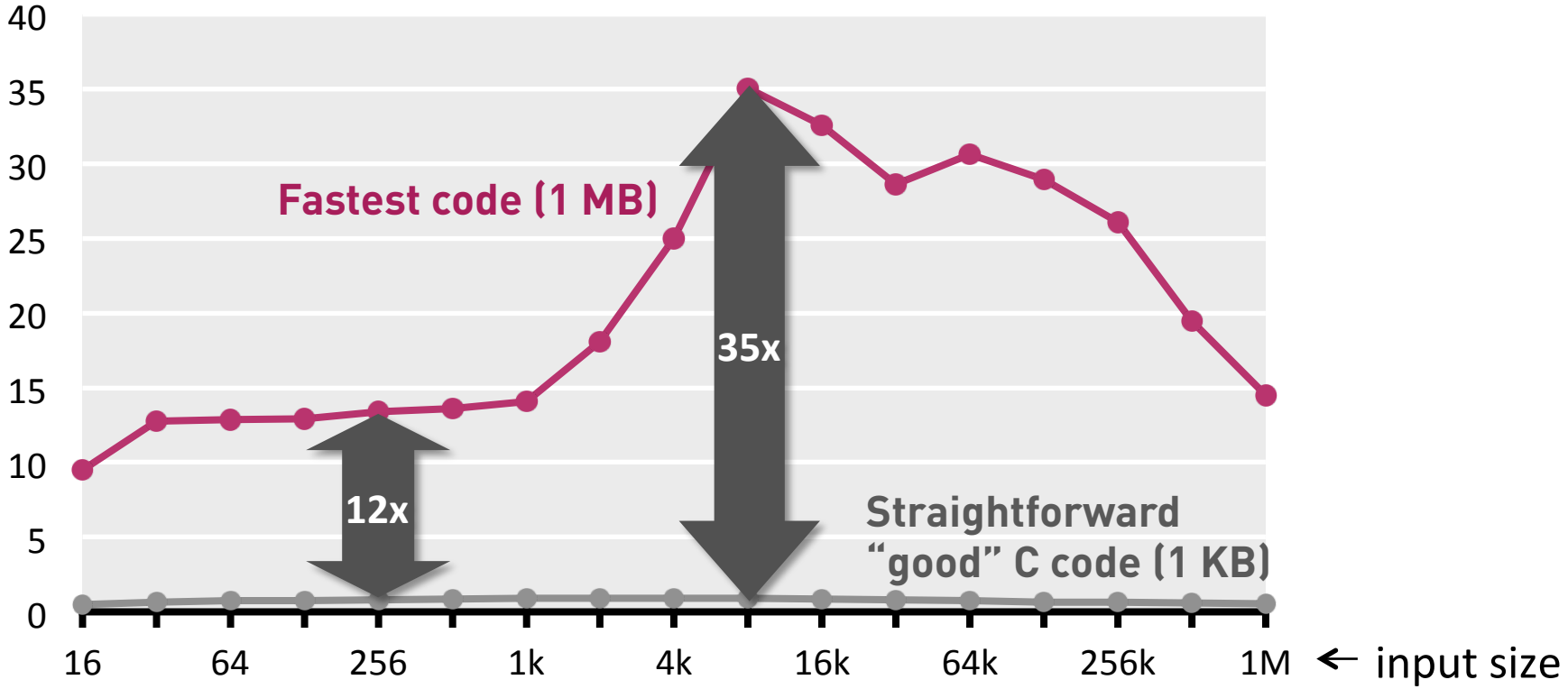
DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)



Vendor compiler, best flags

Example: Discrete Fourier Transform

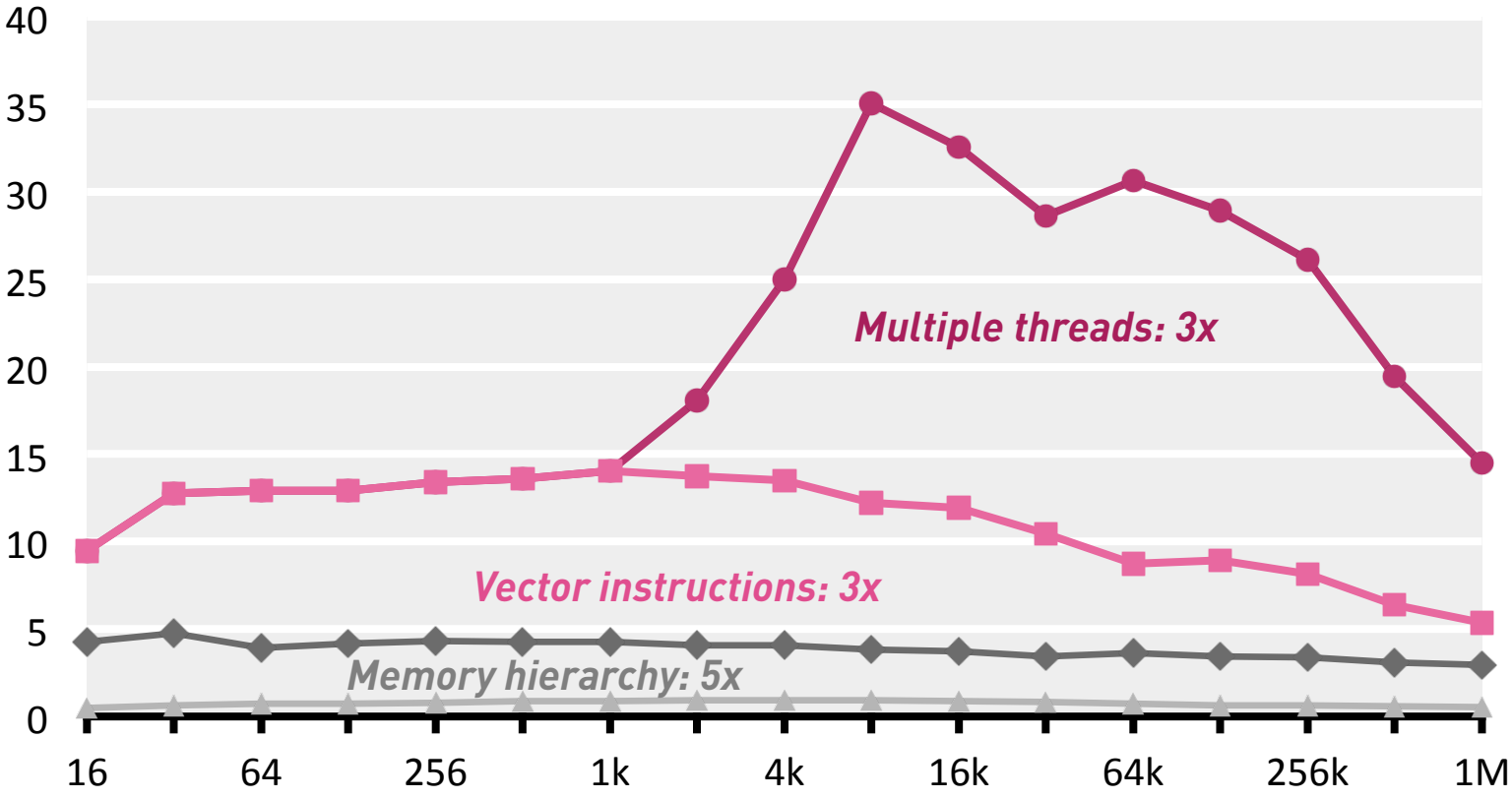
DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)



Vendor compiler, best flags

Roughly same operations count

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



Compiler doesn't do the job

Doing by hand = restructure algorithm for locality & parallelism, handle choices, choose proper code style, use vector intrinsics, ...
= nightmare

Model predictive control

Eigenvalues

LU factorization

Optimal binary search organization

Image color conversions

Image geometry transformations

Enclosing ball of points

Metropolis algorithm, Monte Carlo

Seam carving

SURF feature detection

Submodular function optimization

Graph cuts, Edmond-Karps Algorithm

Gaussian filter

Black Scholes option pricing

Disparity map refinement

Singular-value decomposition

Mean shift algorithm for segmentation

Stencil computations

Displacement based algorithms

Motion estimation

Multiresolution classifier

Kalman filter

Object detection

IIR filters

Arithmetic for large numbers

Optimal binary search organization

Software defined radio

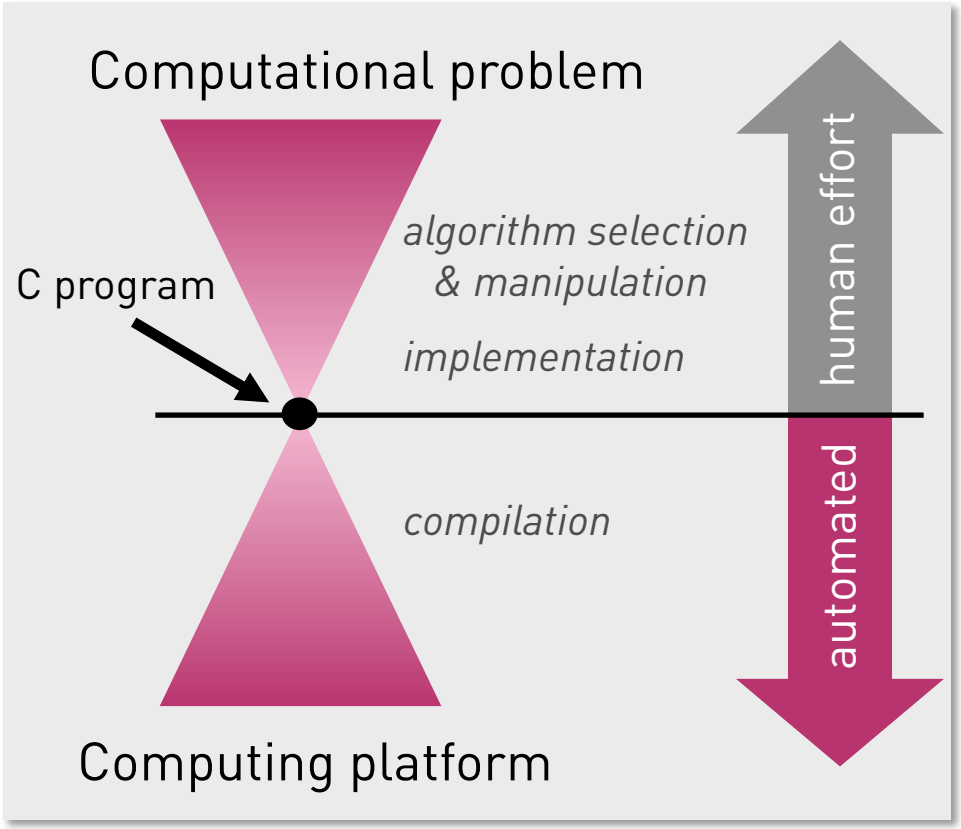
Shortest path problem

Feature set for biomedical imaging

Biometrics identification

Same for (almost) all computational problems:
Straightforward code is highly suboptimal

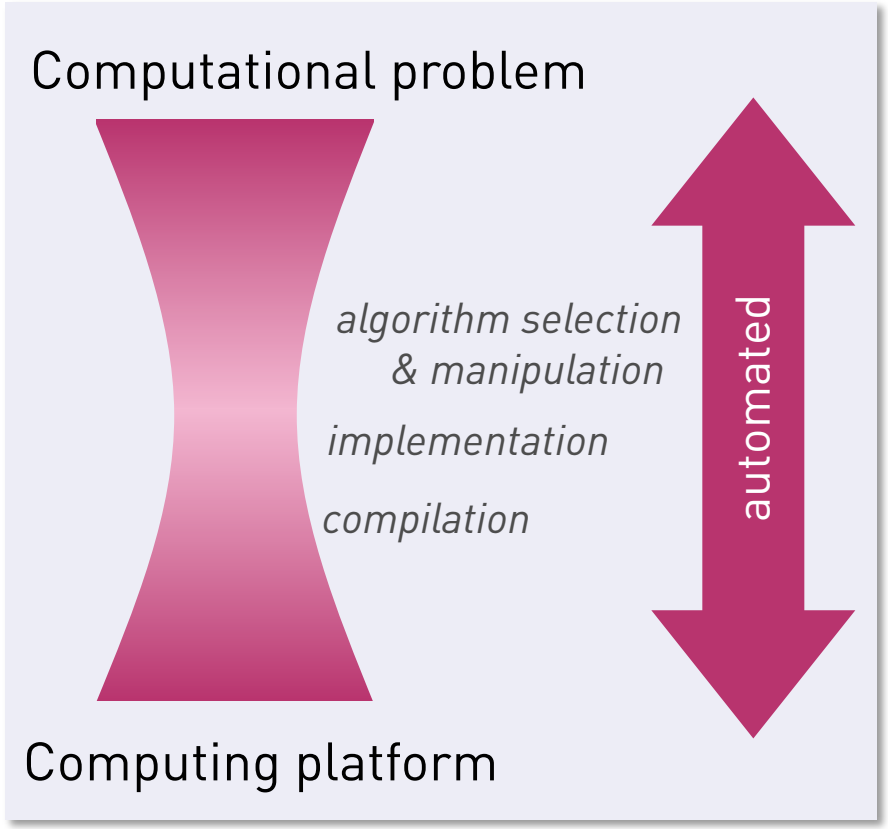
Current



C code is a singularity:

- Compiler has no access to high level information
- No structural optimization
- No evaluation of choices

Future



Challenge: conquer the high abstraction level for ***more/complete automation***

DSLs!

Example: Spiral

Computer Generation of Fast DFTs

www.spiral.net

Recursive algorithms expressed as rules in mathematical, internal DSL

$$\mathbf{DFT}_n \rightarrow (\mathbf{DFT}_k \otimes I_m) T^n_m (I_k \otimes \mathbf{DFT}_m) L^n_k$$
$$\mathbf{DFT}_n \rightarrow P^{\top}_{k/2,2m} \left(\mathbf{DFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i C_{2m}\, \mathbf{rDFT}_{2m}(i/k)\right)\right) (\mathbf{RDFT}_k \otimes I_m)$$
$$\mathbf{RDFT}_n \rightarrow (P^{\top}_{k/2,m} \otimes I_2) \left(\mathbf{RDFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i D_{2m}\, \mathbf{rDFT}_{2m}(i/k)\right)\right) (\mathbf{RDFT}_k \otimes I_m)$$
$$\mathbf{rDFT}_{2n}(u) \rightarrow L^{2n}_m \left(I_k \otimes_i \mathbf{rDFT}_{2m}((i+u)/k)\right) (\mathbf{rDFT}_{2k}(u) \otimes I_m)$$

Recursive combination yields many choices

Example: Spiral

Transform DFT_8

Decomposition rules

Algorithm *(DSL 1)*

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

parallelization vectorization

Algorithm *(DSL 2)*

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

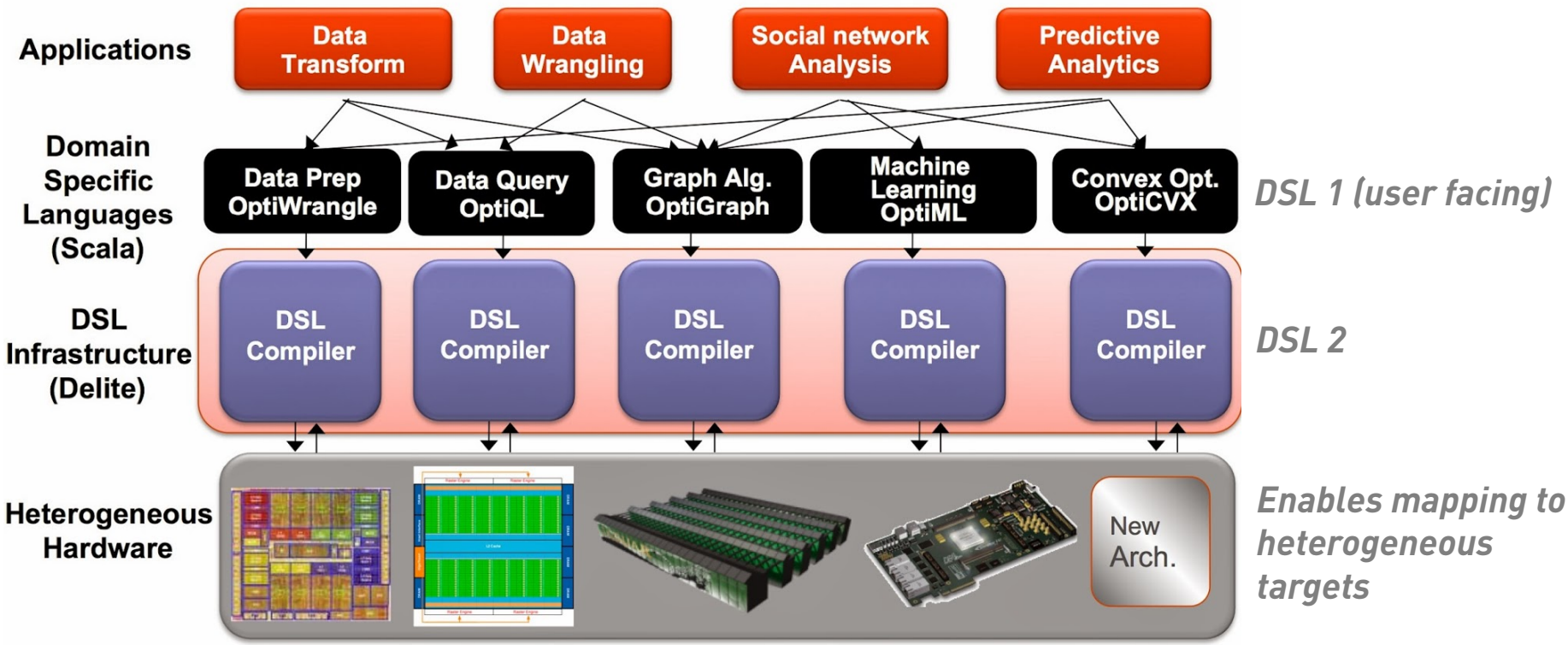
locality optimization

C Program + ext.

```
void sub(double *y, double *x) {
double f0, f1, f2, f3, f4, f7, f8, f10, f11;
...
t282 = _mm_addsub_ps(t268, U247);
t283 = _mm_add_ps(t282, _mm_addsub_ps(U247, _mm_shuffle_ps(t275,
t284 = _mm_add_ps(t282, _mm_addsub_ps(U247, _mm_sub_ps(_mm_setze
s217 = _mm_addsub_ps(t270, U247);
s219 = _mm_shuffle_ps(t278, t280, _MM_SHUFFLE(1, 0, 1, 0));
s220 = _mm_shuffle_ps(t278, t280, _MM_SHUFFLE(3, 2, 3, 2));
s221 = _mm_shuffle_ps(t283, t285, _MM_SHUFFLE(1, 0, 1, 0));
...
< many more lines>
```

+ Search or Learning for Choices

Example: Delite



Generating Fast Database Code with DBLAB

Maps query/transaction workloads to embedded Scala DSL.

DSL compiler with a rich set of domain-specific code transformers (data layout transformations, data structure specialization, index introduction, materialization decisions, ...)

Uses code transformations on multiple abstraction levels. Successive lowering phases.

Generates fast C code