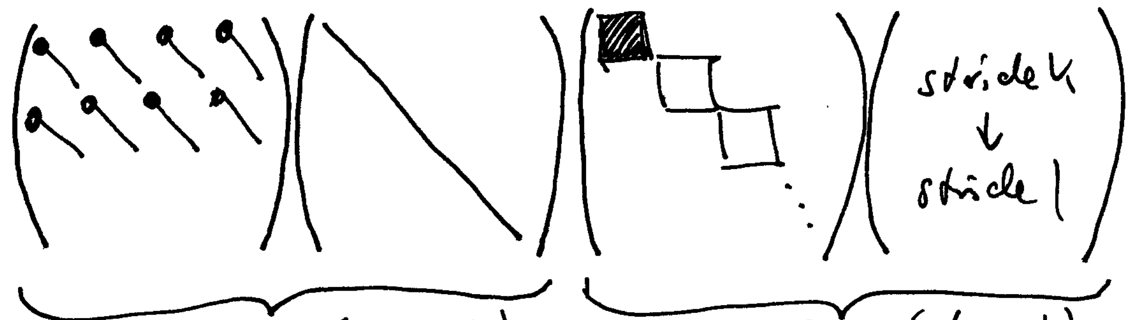
Locality of data access

1.) Choose recursive FFT, not iterative FFT

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

(schematic) =



fuse (stage 2) fuse (stage 1)

compute m many
$DFT_k \cdot D$
part of twiddles

- writes to same locations it reads from
$\Rightarrow$ in place

- stride as parameter
- writes to different locations than it reads from
$\Rightarrow$ out-of-place

DFTtwiddle(k, *x, *t, stride)
size, input = output vector, twiddles

DFTrec(m, *x, *y, instride, outstride)
size, input vector, output vector

cannot handle arbitrary recursion
$\Rightarrow$ in FFTW a base case

the interface handles arbitrary recursions

<u>Pseudocode:</u>

```
DFT(n, *x, *y) = DFTrec(n, *x, *y, 1, 1)
   for i = 0:k-1
DFTrec(m, *x+i, *y+im, k, 1)          } stage 1: recurse until base case
   for j = 0:m-1
DFTtwiddle(k, *y+j, *t_j, m)          } stage 2: base case
```

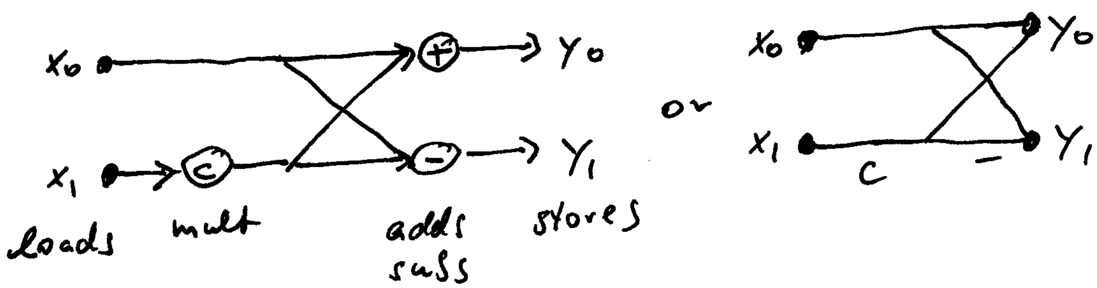
$*t_j$: explained later

$$y \xleftarrow{\text{stage 2}} y \xleftarrow{\text{stage 1}} x$$

DAG example

formula: $DFT_2 \cdot diag(1, c)$

DAG:



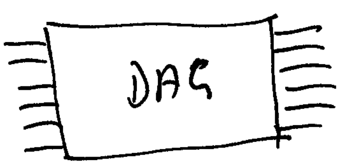
Simplifications

$t_5 = 0 \cdot t_2 \quad \rightarrow \quad t_5 = 0$

$$\begin{aligned} t_3 &= \dots \\ t_4 &= 1 \cdot t_3 \\ t_5 &= t_4 + t_1 \end{aligned} \quad \rightarrow \quad \begin{aligned} t_3 &= \dots \\ t_5 &= t_3 + t_1 \end{aligned}$$

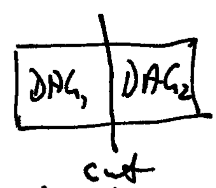
$$\begin{aligned} t_3 &= 2t_1 \\ t_4 &= -2t_2 \\ t_5 &= t_0 + t_4 \end{aligned} \quad \rightarrow \quad \begin{aligned} t_3 &= 2t_1 \\ t_4 &= 2t_2 \\ t_5 &= t_0 - t_4 \end{aligned}$$

Scheduling

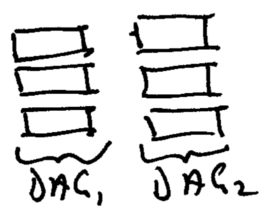


$\longrightarrow$ C code (sequential)

Step 1: cut DAG in middle (how?)



Step 2: the two pieces decompose into independent DAGs

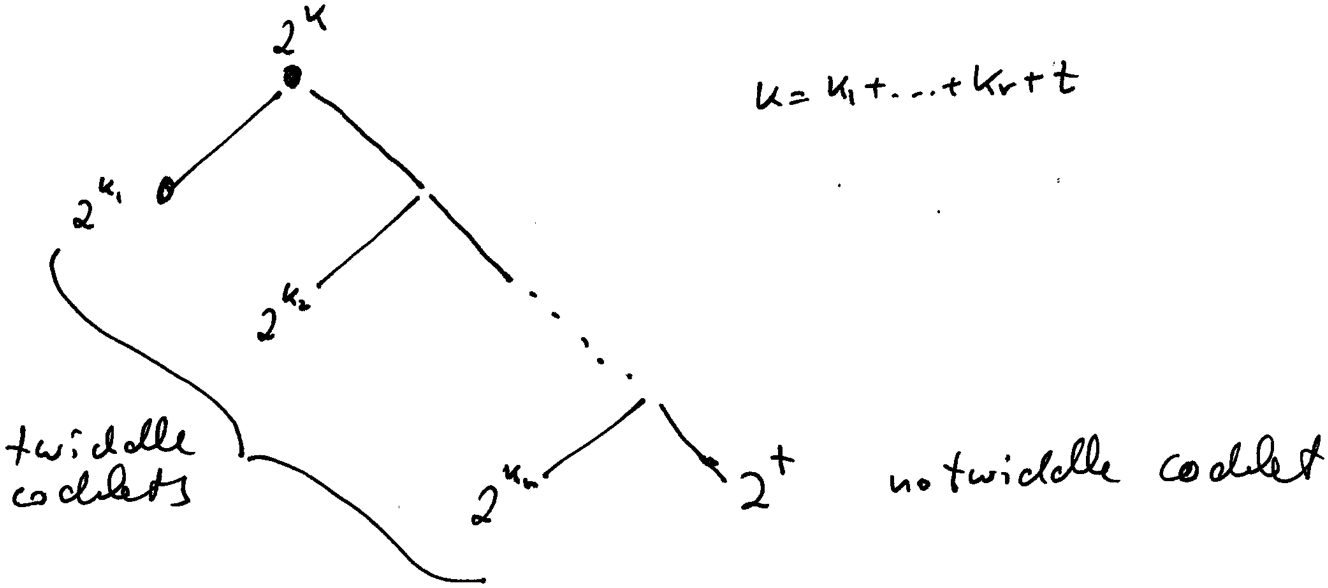


Step 3, schedule those recursively using the same method

Adaptivity

Space of algorithms considered by FFTW 2.x

$n = 2^k$



Best choice found by dynamic programming.