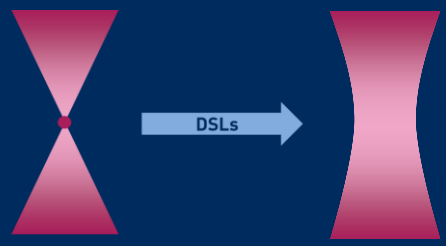


LGen
A Basic Linear Algebra Compiler

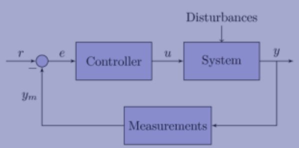
Daniele Spampinato
Markus Püschel

Computer Science
ETH zürich

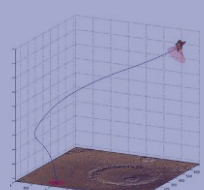
SPIRAL
www.spiral.net

Linear algebra: Central to many domains

Control systems



Optimization algorithms



Source: rain.aa.washington.edu

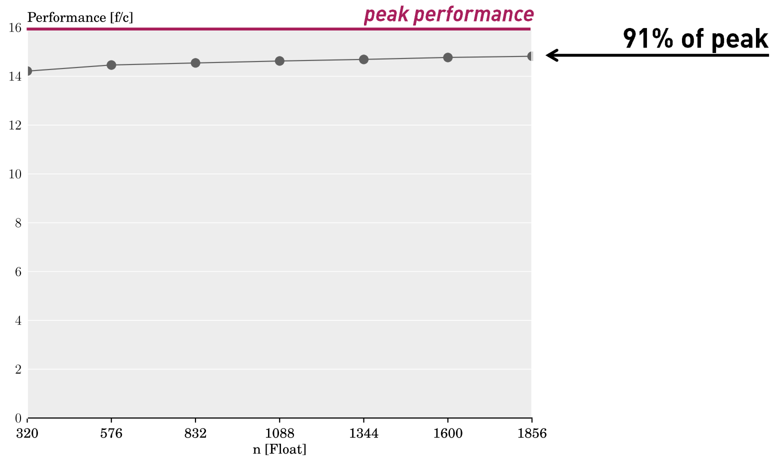
Computer Graphics

Source: disneyresearch.com

Quoting *App performance* section
"The frame rate of 12 FPS, however, is not enough to handle sudden motions. In future work, optimisation of implementation would improve the frame rate and provide better user experience."

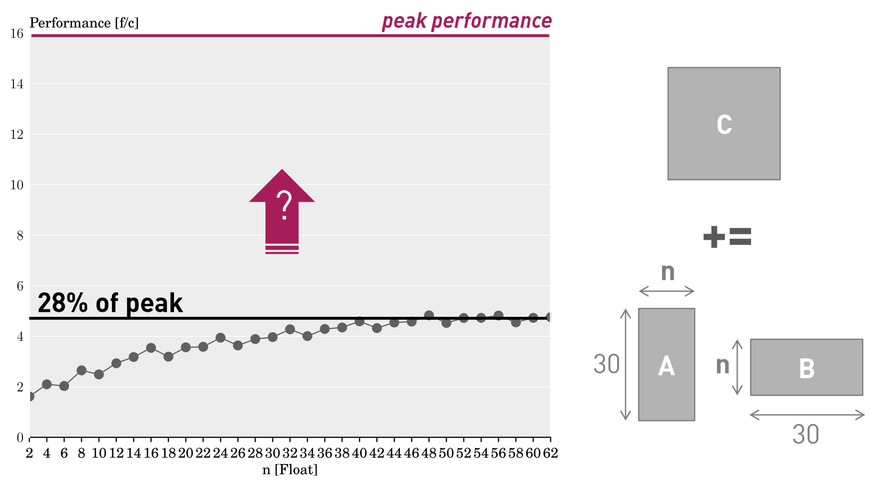
Library performance for sgemm (C += AB)

Intel MKL on Intel Core i7 CPU (AVX)



A closer look at small problem sizes

Intel MKL on Intel Core i7 CPU (AVX)



Bridging the gap with DSLs

Kalman Filter

Predict

$$X = Ax_{k-1} + Bu_{k-1}$$

$$p = APA^T + Q$$

Update

$$K = pH^T \left(HpH^T + R\right)^{-1}$$

$$x_k = X + K(z - HX)$$

$$Px_k = (I - KH)p$$

Goal

```
void kf(double const * A, ...) {
  __m256d t0, …;

  a0 = _mm256_loadu_pd(A);
  a1 = _mm256_load_sd(A + 4);
  ...
  m0 = _mm256_mul_pd(a0, x0);
  ...
  h0 = _mm256_hadd_pd(m0, m1);
  ...
  p =
_mm256_permute2f128_pd(...);
  b = _mm256_blend_pd(t6, t8);
  ...
  _mm256_storeu_pd(X, r0);
  ...

}
```

LGen: Base System

Basic Linear Algebra Computations (BLACs)

Examples:

$$y = Ax$$

$$C = \alpha AB^T + \beta C$$

$$\gamma = x^T(A + B)y + \delta$$

Composed of:

Scalars, vectors, and matrices

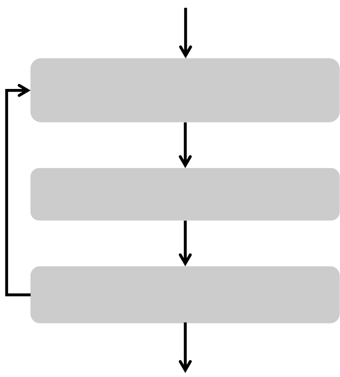
Operators:

- *Addition*
- *Scalar multiplication*
- *Matrix multiplication*
- *Transposition*

All input and output vectors and matrices have a fixed size

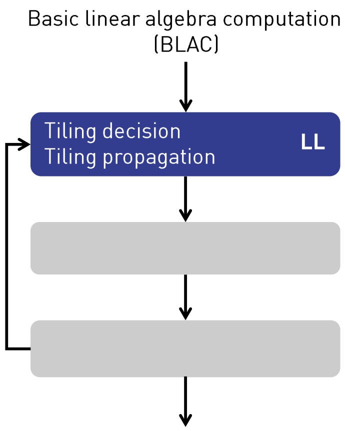
LGen: A basic linear algebra compiler

Basic linear algebra computation (BLAC)



$y = Ax \leftarrow$ **A** is 2 x 3, **x** is 3 x 1

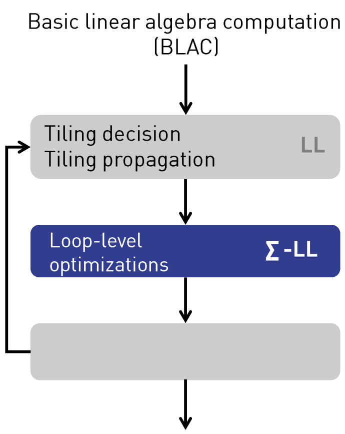
LGen: A basic linear algebra compiler



$$y = Ax \leftarrow$$ **A** is 2 x 3
x is 3 x 1

$$[y = Ax]_{2,1}$$

LGen: A basic linear algebra compiler

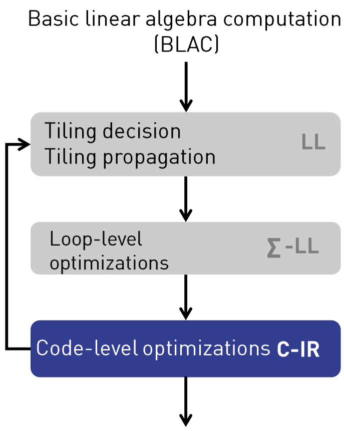


$$y = Ax \leftarrow$$ **A** is 2 x 3
x is 3 x 1

$$[y = Ax]_{2,1}$$

$$y = \sum_{i,j} [i] \left(A[i,j] x[j] \right)$$

LGen: A basic linear algebra compiler



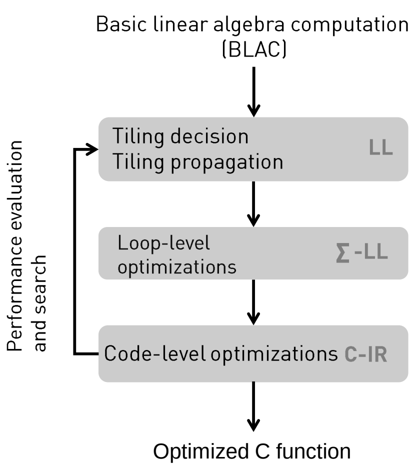
$y = Ax$ ← **A** is 2 x 3
x is 3 x 1

$[y = Ax]_{2,1}$

$y = \sum_{i,j} [i]\,(A[i,j]x[j])$

…
Mov (mmMulPs A[0,0], x[0,0]), t[0,0]
…

LGen: A basic linear algebra compiler



$y = Ax$ ← **A** is 2x3
x is 3x1

$[y = Ax]_{2,1}$

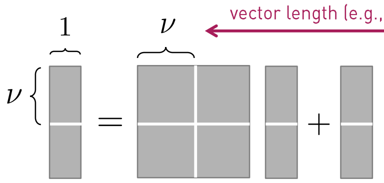
$y = \sum_{i,j} [i]\,(A[i,j]x[j])$

…
Mov (mmMulPs A[0,0], x[0,0]), t[0,0]
…

```
for(int i = … ) {
 …
   t = _mm_mul_ps(a, x);
 …
}
```

Vector code generation: Basic Idea

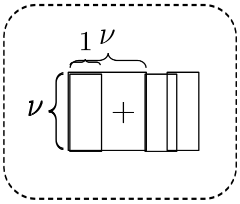
vector length (e.g., 1 for scalar float/double, 4 for SSE float)



Goal: express the computation in terms of v-BLACs

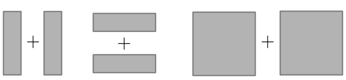
$$[y]_{\nu,1} = [A]_{\nu,\nu}[x]_{\nu,1} + [y]_{\nu,1}$$

$$y = \sum_{i,j}[i]\left(A[i,j]x[j] + y[i]\right)$$

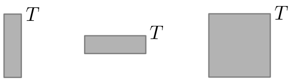


v-BLACs: Vectorization building blocks

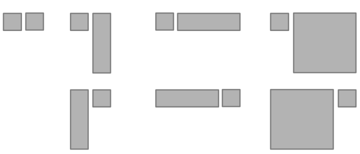
Addition (3 v-BLACs)



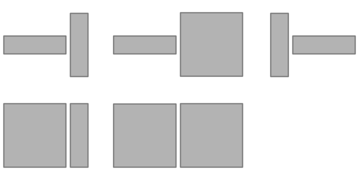
Transposition (3 v-BLACs)



Scalar Multiplication (7 v-BLACs)



Matrix Multiplication (5 v-BLACs)



18 cases implemented once for every ISA

The importance of structures

Kalman Filter

Predict

$x_k = Ax_{k-1} + Bu_k$

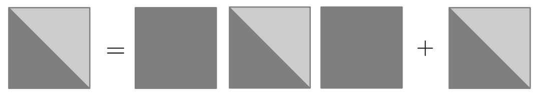
$P_k = AP_{k-1}A^T + Q$

Update

$K = P_kH^T\left(HP_kH^T + R\right)^{-1}$

$x_k = x_k + K\left(z_k - Hx_k\right)$

$P_k = (I - KH)P_k$



The importance of structures

Kalman Filter

Predict

$x_k = Ax_{k-1} + Bu_k$

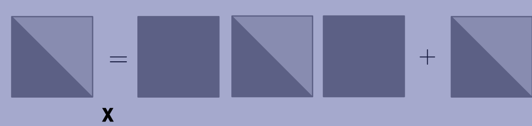
$P_k = AP_{k-1}A^T + Q$

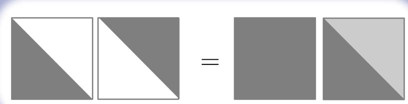
Update

$K = P_kH^T\left(HP_kH^T + R\right)^{-1}$

$x_k = x_k + K\left(z_k - Hx_k\right)$

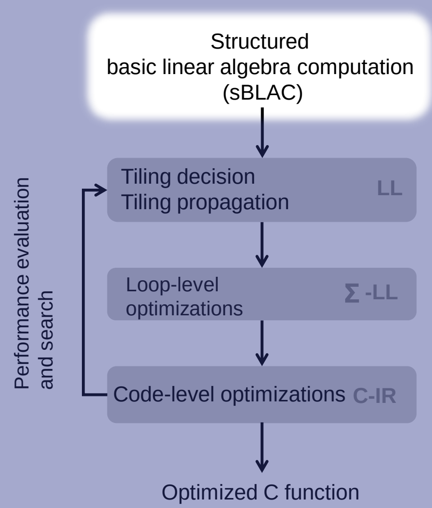
$P_k = (I - KH)P_k$





Extending LGen with Structures

Code generation including structures



$y = Ax$ ← **A** is 2x3 and **x** is 3x1

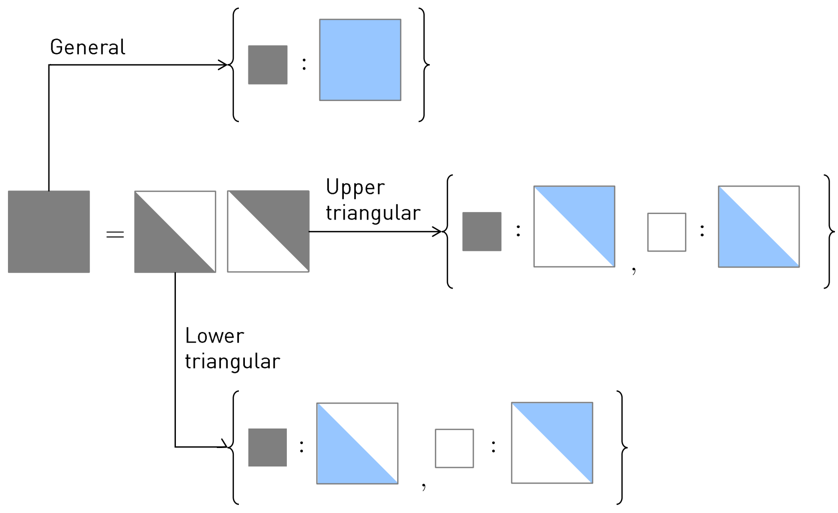
$[y = Ax]_{2,1}$

$y = \sum_{i,j} [i]\,(A[i,j]x[j])$

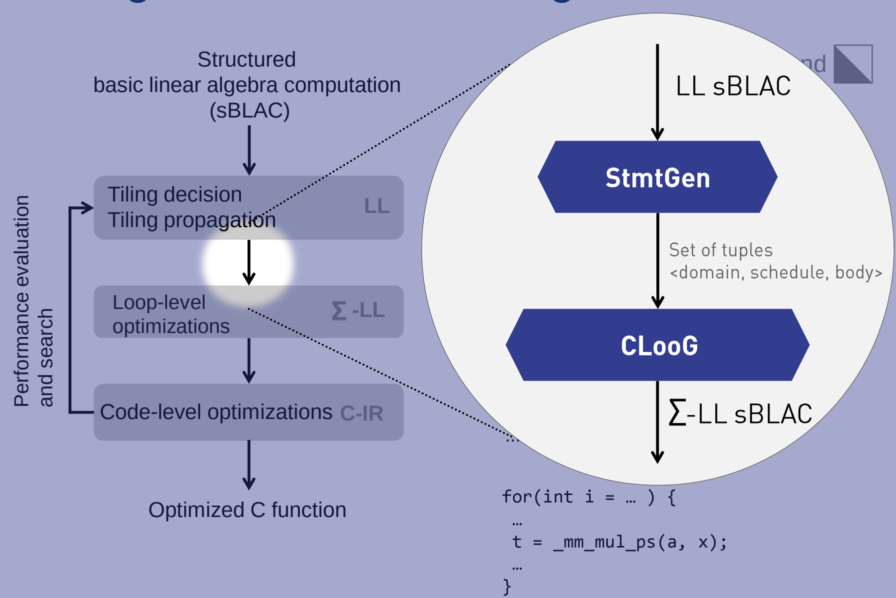
…
Mov (mmMulPs A[0,0], x[0,0]), t[0,0]
…

```
for(int i = … ) {
 …
 t = _mm_mul_ps(a, x);
 …
}
```

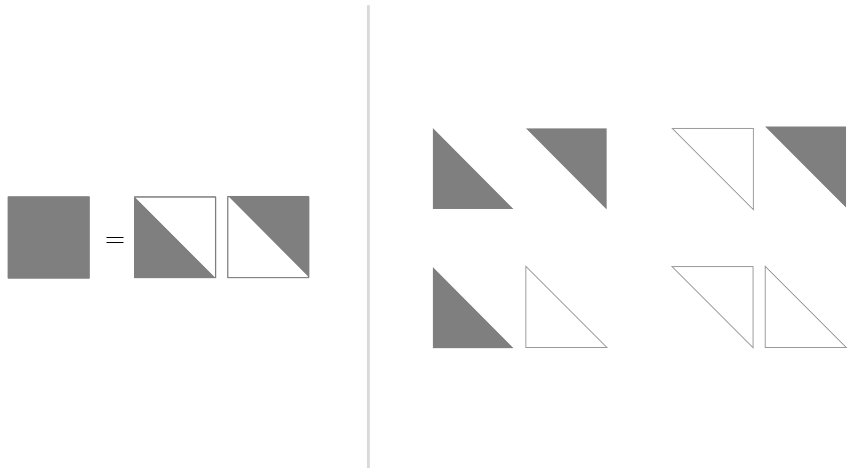
Structured matrices representation



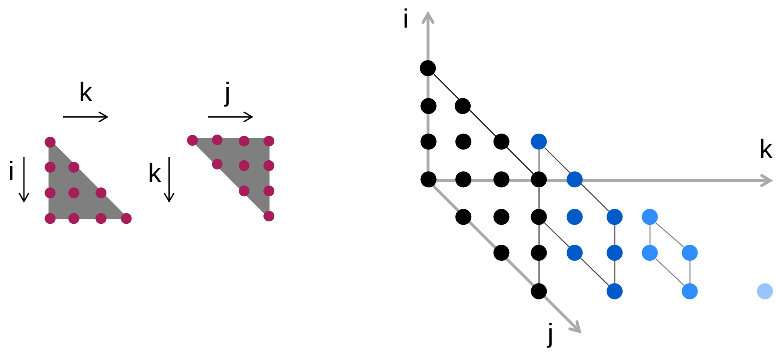
Code generation including structures



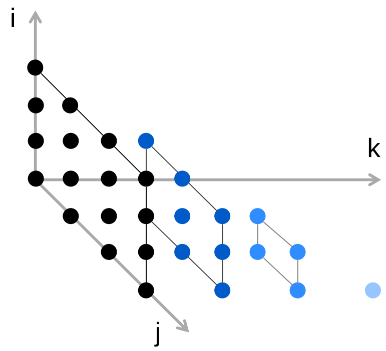
From LL to Σ-LL



From LL to Σ-LL

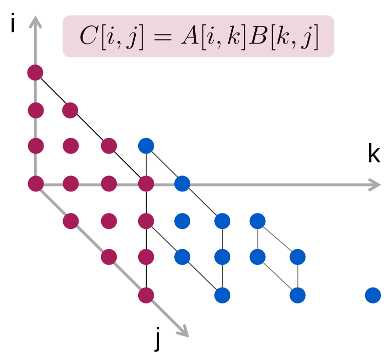


From LL to Σ-LL



From LL to Σ-LL

$C[i,j] = A[i,k]B[k,j]$



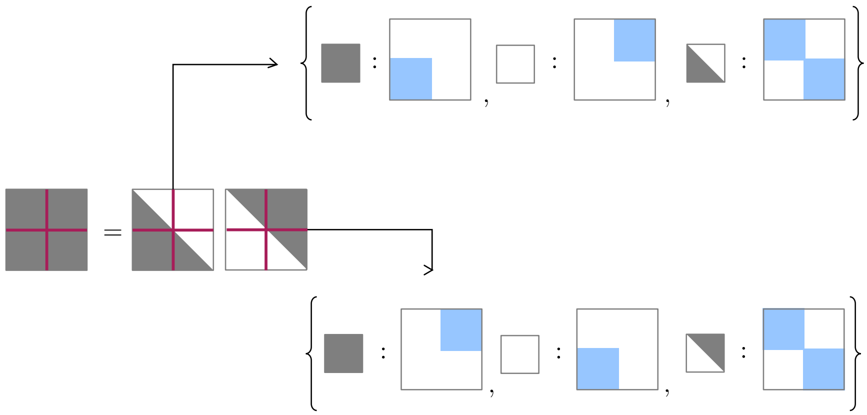
CLooG

$$C = \sum_{i=0}^{3}\sum_{j=0}^{3}[i,j]\,(A[i,0]B[0,j]) + \sum_{k=1}^{3}\sum_{i=k}^{3}\sum_{j=k}^{3}[i,j]\,(A[i,k]B[k,j])$$

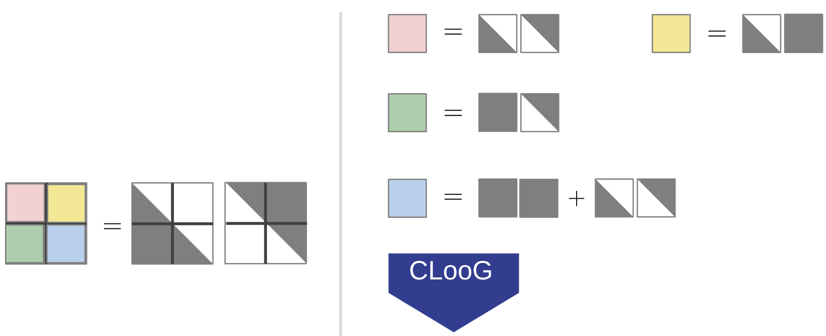
$C[i,j] = C[i,j] + A[i,k]B[k,j]$

Scattering relation built based on optimization models (e.g., Goto model)

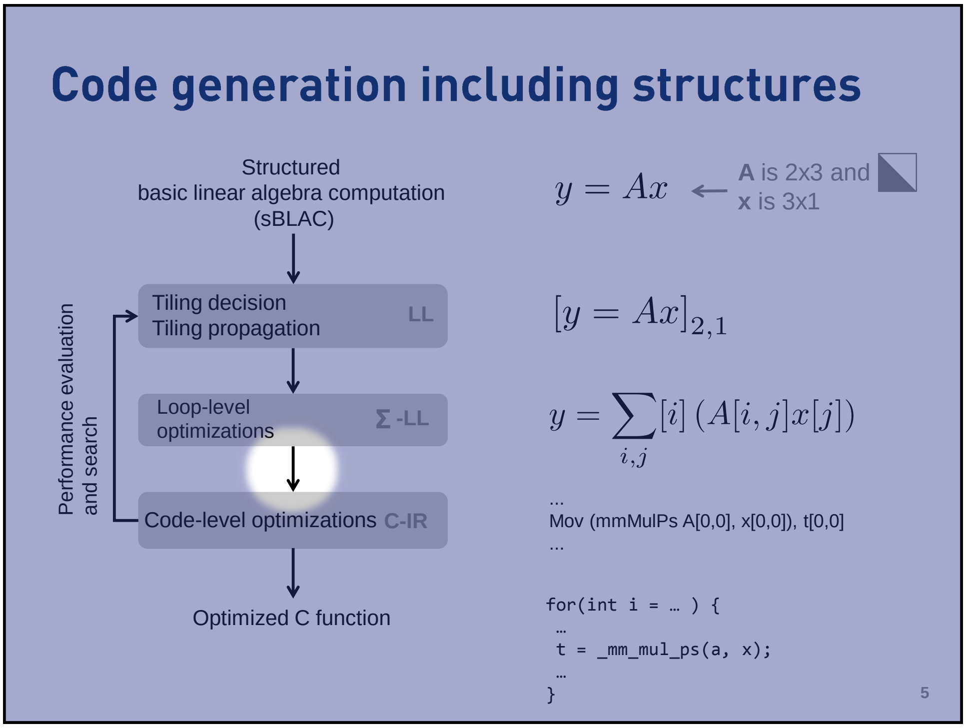
Representing structured tiled matrices



From LL to Σ-LL



$$
\begin{aligned}
C = {} & [0,0](A[0,0]B[0,0]) \\
& +[0,2](A[0,0]B[0,2]) \\
& +[2,0](A[2,0]B[0,0]) \\
& +[2,2](A[2,0]B[0,2] + A[2,2]B[2,2])
\end{aligned}
$$


Code generation including structures
Structured
basic linear algebra computation
(sBLAC)
Tiling decision
Tiling propagation
LL
Loop-level
optimizations
Σ -LL
Code-level optimizations C-IR
Optimized C function
Performance evaluation
and search
$y = Ax$
A is 2x3 and
x is 3x1
$[y = Ax]_{2,1}$
$y = \sum_{i,j} [i]\,(A[i,j]x[j])$
…
Mov (mmMulPs A[0,0], x[0,0]), t[0,0]
…
for(int i = …) {
…
t = _mm_mul_ps(a, x);
…
}
5


Experiments

Experimental settings

Intel Core i7 (Sandy Bridge)

Ubuntu 14.04 with Linux 3.13

Double computations with AVX

Warm cache scenario (32 kB L1 D-cache, 256 kB L2 cache)

Four competitors:

LGen w/ and w/0 structures support

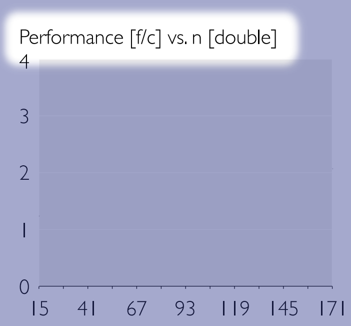
Intel MKL 11.2

Naïve code (non optimized double/triple loop code)

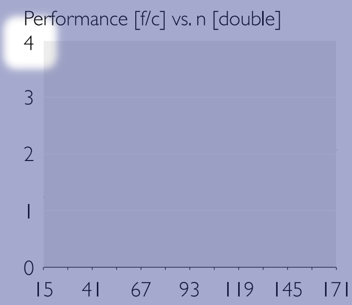
All kernels compiled with icc 15 w/ flags:

```
-O3 -xHost -fargument-noalias -fno-alias
```

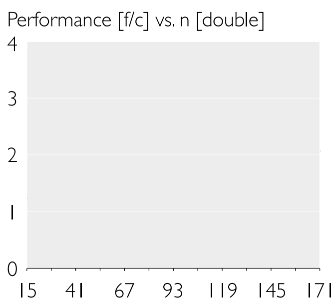
Plotting

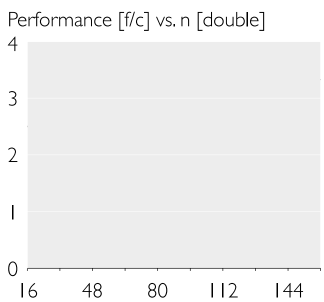


Plotting

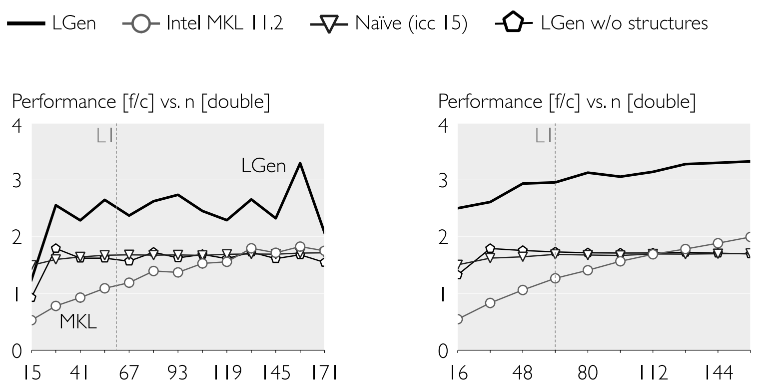


Plotting



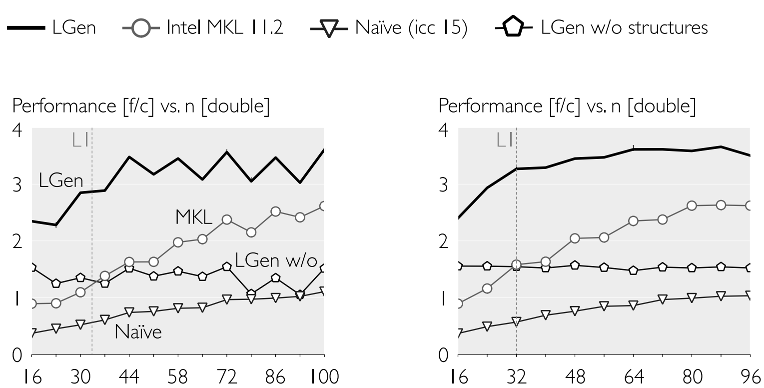


BLAS category: dsyrk



$S_u = AA^T + S_u, \qquad A \in \mathbb{R}^{n\times 4}$

BLAS-like category

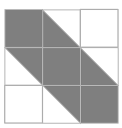


$A = LU + S_l, \qquad L, U \in \mathbb{R}^{n\times n}$

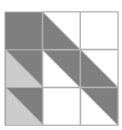
More general structures
&
future work

Approach extensibility

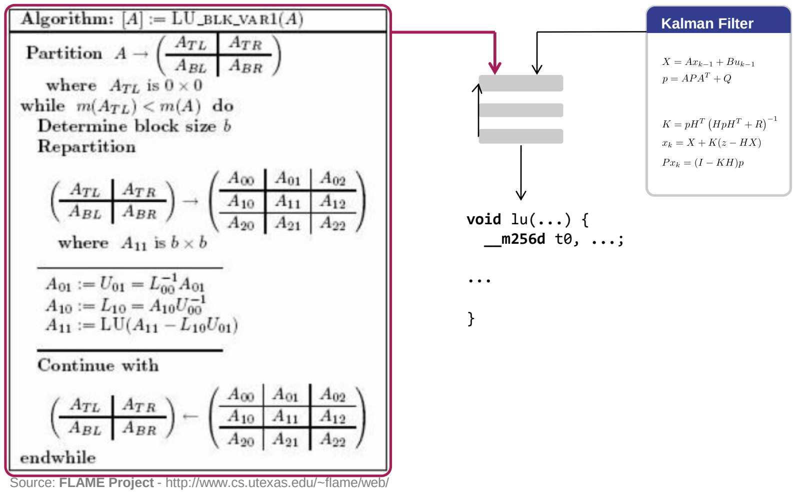
Other important structures, e.g., banded matrices



Or other combined structures



Support for higher-level functionalities

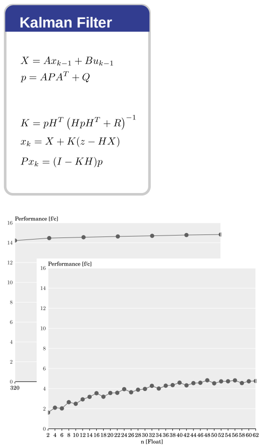


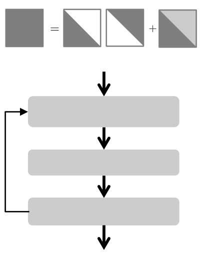
Source: **FLAME Project** - http://www.cs.utexas.edu/~flame/web/

Connecting with Cl1ck (Fabregat-Traver, Bientinesi)

Conclusion

Our Approach

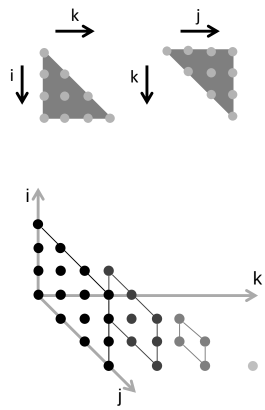




```
void kernel(...) {
  __m256d t0, ...;

...

}
```



spiral.net/software/lgen.html