

## Case study:

Tall & Skinny Matrix-Transpose Times  
Tall & Skinny Matrix (TSMTTSM)  
Multiplication



# TSMTTSM Multiplication

- **Block of vectors** → Tall & Skinny Matrix (e.g.  $10^7 \times 10^1$  dense matrix)
- Row-major storage format (see SpMVM)
- Block vector subspace orthogonalization procedure requires, e.g., computation of scalar product between vectors of two blocks

- → **TSMTTSM** Multiplication

$$C = \alpha A^T * B + \beta C$$

Assume:  $\alpha = 1$ ;  $\beta = 0$

# TSMITTSM Multiplication

General rule for dense matrix-matrix multiply: Use vendor-optimized GEMM, (e.g., Intel MKL<sup>1</sup>):

$$C_{mn} = \sum_{k=1}^K A_{mk} B_{kn}, \quad m = 1..M, n = 1..N$$

System	P <sub>peak</sub> [GF/s]	b <sub>s</sub> [GB/s]	Size	Perf.	Efficiency
Intel Xeon E5 2660 v2 10c@2.2 GHz	176 GF/s	52 GB/s	SQ	160 GF/s	91%
			TS	16.6 GF/s	6%
Intel Xeon E5 2697 v3 14c@2.6GHz	582 GF/s	65 GB/s	SQ	550 GF/s	95%
			TS	22.8 GF/s	4%

double

complex double

TS@MKL:  
Good or bad?

Matrix sizes:

Square (SQ): M=N=K=15,000

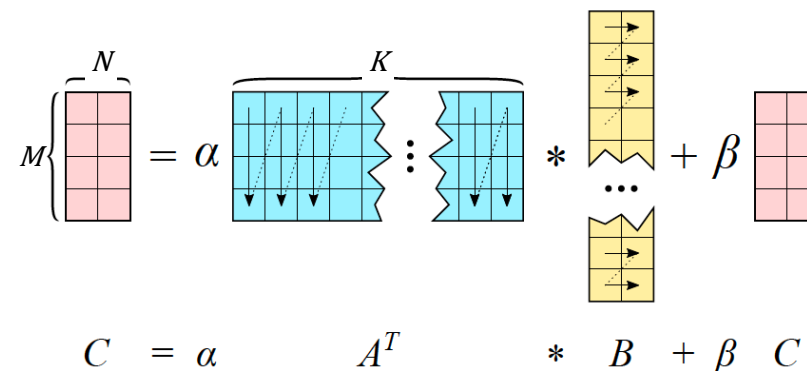
Tall&Skinny (TS): M=N=16 ; K=10,000,000

<sup>1</sup>Intel Math Kernel Library (MKL) 11.3

# TSMTTSM Roofline model

Computational intensity

$$I = \frac{\text{\#flops}}{\text{\#bytes (slowest data path)}}$$



Optimistic model (minimum data transfer) assuming  $M = N \ll K$  and double precision:

$$I_d \approx \frac{2KMN}{8(KM + KN)} \frac{F}{B} = \frac{MF}{8B}$$

complex double:

$$I_z \approx \frac{8KMN}{16(KM + KN)} \frac{F}{B} = \frac{MF}{4B}$$

# TSMTTSM Roofline performance prediction

Now choose  $M = N = 16 \rightarrow I_d \approx \frac{16 F}{8 B}$  and  $I_z \approx \frac{16 F}{4 B}$ , i.e.  $B_d \approx 0.5 \frac{B}{F}$ ,  $B_z \approx 0.25 \frac{B}{F}$

Intel Xeon E5 2660 v2 ( $b_s = 52 \frac{GB}{s}$ )  $\rightarrow P = 104 \frac{GF}{s}$  (double)

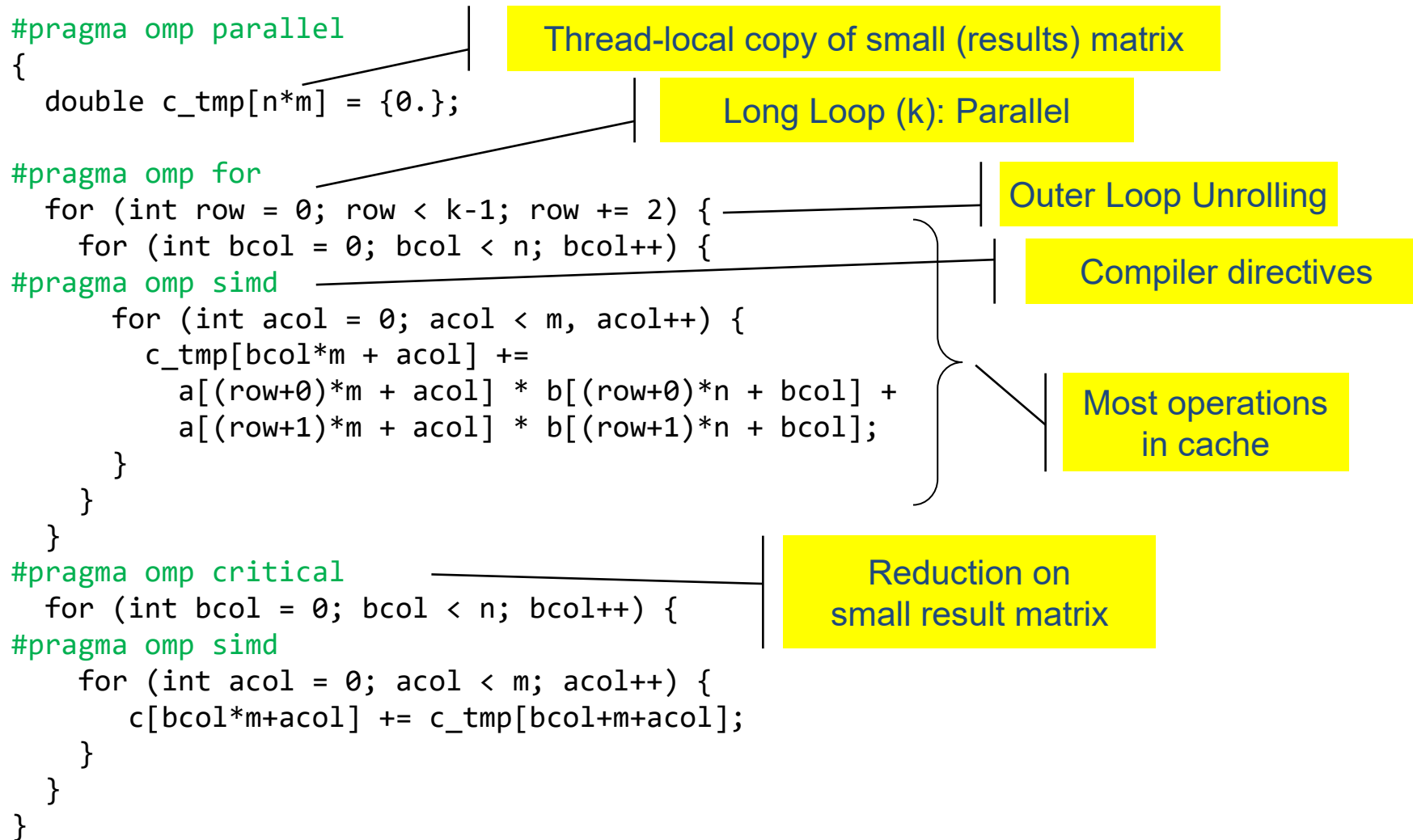
Measured (MKL):  $16.6 \frac{GF}{s}$

Intel Xeon E5 2697 v3 ( $b_s = 65 \frac{GB}{s}$ )  $\rightarrow P = 240 \frac{GF}{s}$  (double complex)

Measured (MKL):  $22.8 \frac{GF}{s}$

$\rightarrow$  Potential speedup: 6–10x vs. MKL

# Can we implement a better TSMTTSM kernel than Intel?

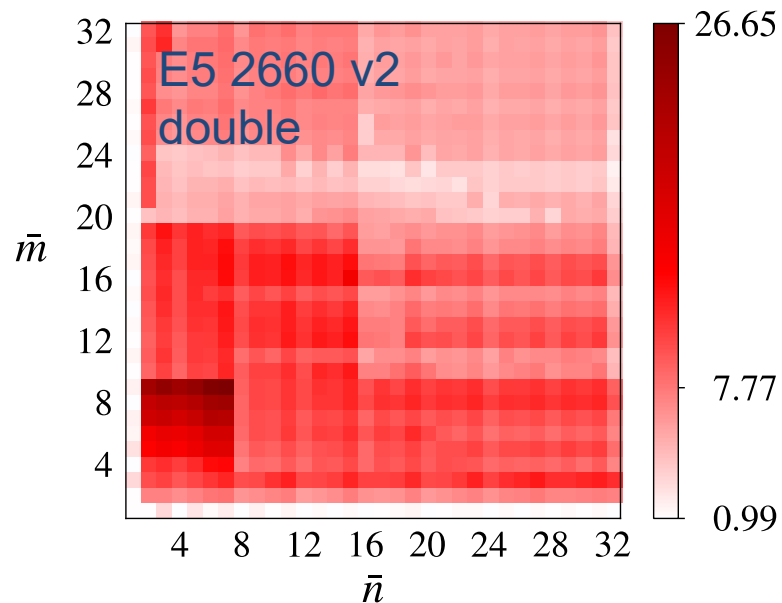


Not shown: Inner Loop boundaries (n,m) known at compile time (kernel generation), k assumed to be even

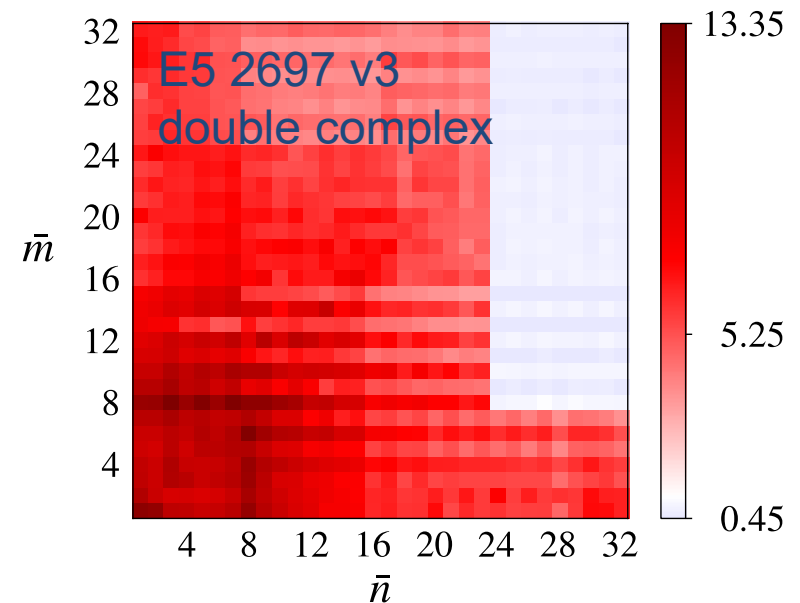
# TSMITTSM MKL vs. “hand crafted” (OPT)

TS: M=N=16 ; K=10,000,000

System	$P_{\text{peak}} / b_s$	Version	Performance	RLM Efficiency
Intel Xeon E5 2660 v2 10c@2.2 GHz	176 GF/s 52 GB/s	TS OPT	98 GF/s	94 %
		TS MKL	16.6 GF/s	16 %
Intel Xeon E5 2697 v3 14c@2.6GHz	582 GF/s 65 GB/s	TS OPT	159 GF/s	66 %
		TS MKL	22.8 GF/s	9.5 %



Speedup  
vs. MKL:  
5x – 25x



# TSMTTSM conclusion

---

- Typical example of **model-guided optimization**
- “Invisible”  $P_{\max}$  ceiling with Intel MKL (probably wrong loop parallelized)
- Hand-coded implementation ran much closer to limit
  
- **Caveat:**  
This is to exemplify the method; current MKL versions might have improved!