Case study:
tall & skinny matrix-matrix (TSMM) multiplication

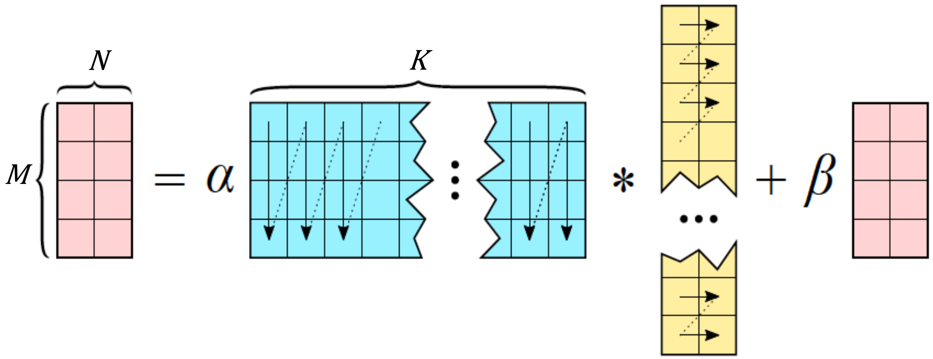
Erlangen Regional Computing Center

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

High Performance Computing

TSMM multiplication

- Block of vectors → Tall & Skinny Matrix (e.g. 10^7 x 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
- TSMM mutliplication



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

$$\boxed{\begin{array}{l} K \gg N, M \\ \alpha = 1;\ \beta = 0 \end{array}}$$

TSMM multiplication

- General rule for dense matrix-matrix multiply: Use vendor-optimized GEMM, e.g. from Intel MKL[1]:

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

double

System	P_{peak} [GF/s]	b_S [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%

complex double

TS@MKL:
Good or bad?

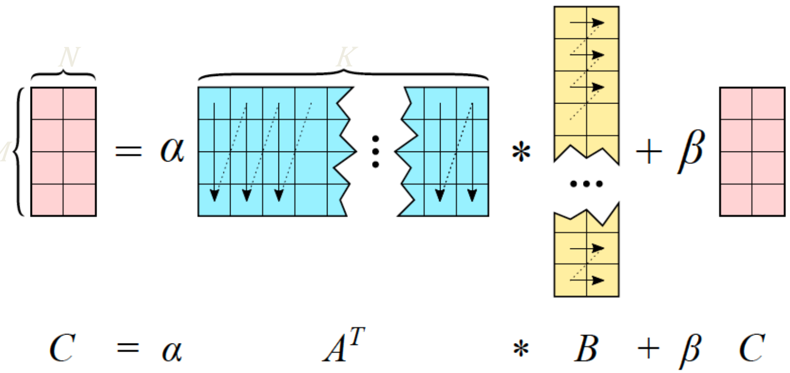
Matrix sizes:
Square (SQ): M=N=K=15,000
Tall&Skinny (TS): M=N=16 ; K=10,000,000

[1]Intel Math Kernel Library (MKL) 11.3

TSMM Roofline model

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

- Computational intensity
- $I = \dfrac{\#\text{flops}}{\#\text{bytes (slowest data path)}}$



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

$K \gg N, M$
$\alpha = 1;\ \beta = 0$

- Optimistic model (minimum data transfer) with $M = N \ll K$

double precision: $I_d \approx \dfrac{2KMN}{8(KM+KN)} \dfrac{\text{F}}{\text{B}} = \dfrac{M}{8} \dfrac{\text{F}}{\text{B}}$

complex double: $I_z \approx \dfrac{8KMN}{16(KM+KN)} \dfrac{\text{F}}{\text{B}} = \dfrac{M}{4} \dfrac{\text{F}}{\text{B}}$

TSMM Roofline model

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

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

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

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

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

→ Potential speedup: 6–10x vs. MKL

Can we implement a better TSMM kernel than Intel?


```
#pragma omp parallel
{
  double c_tmp[n*m] = {0.};

#pragma omp for
  for (int row = 0; row < k-1; row+=2) {
    for (int bcol = 0; bcol < n; bcol++) {
#pragma simd
      for (int acol = 0; acol < m; acol++) {
        c_tmp[bcol*m+acol] +=
          a[(row+0)*m + acol] * b[(row+0)*n + bcol] +
          a[(row+1)*m + acol] * b[(row+1)*n + bcol];
      }
    }
  }

#pragma omp critical
  for (int bcol = 0; bcol < n; bcol++) {
#pragma simd
    for (int acol = 0; acol < m; acol++) {
      c[bcol*m+acol] += c_tmp[bcol*m+acol];
    }
  }
}
```

Thread local copy of small (results) matrix

Long Loop (k): Parallel

Outer Loop Unrolling

Compiler directives

Most operations in cache

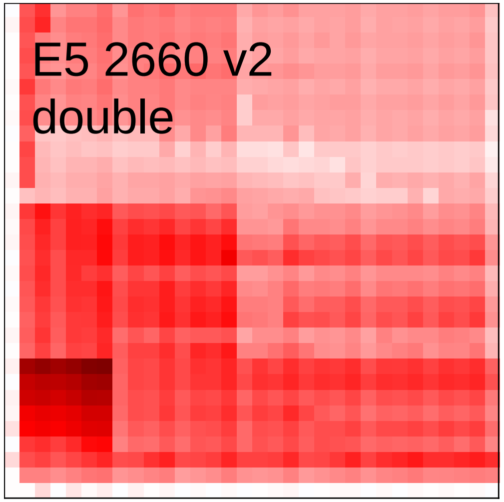
Reduction on small result matrix

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

TSMM: MKL vs. “hand crafted” (OPT)

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

System	P_{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