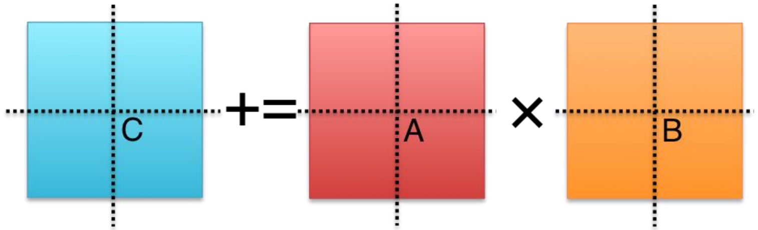
STRASSEN

Strassen's Algorithm Reloaded



Jianyu Huang, Tyler M. Smith,
Greg M. Henry, Robert A. van de Geijn

The University of Texas at Austin, Intel

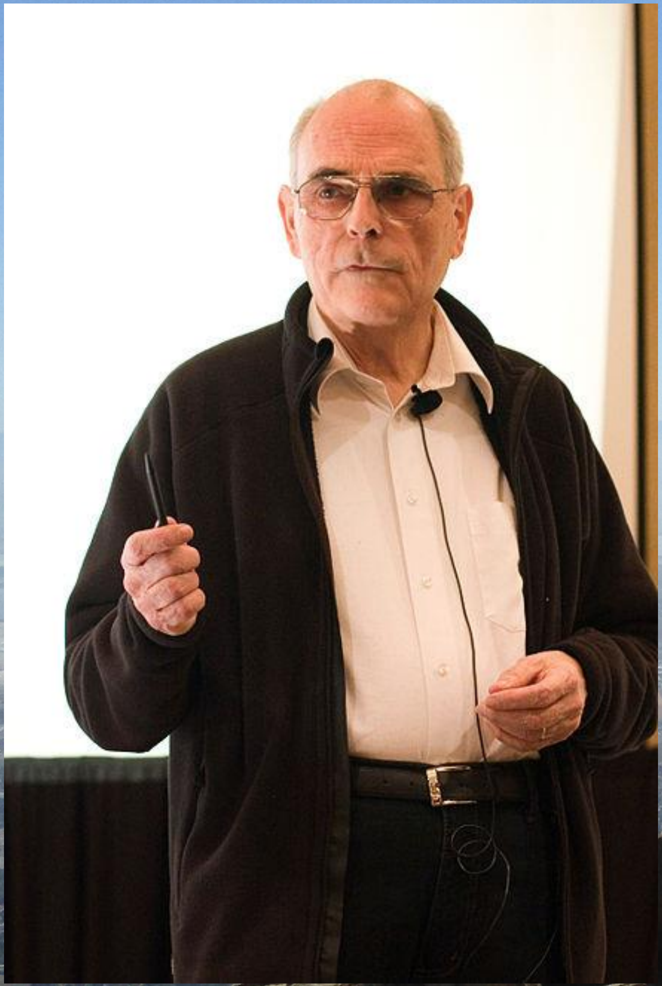
Salt Lake City, UT

November 16th, 2016

*Overlook of the Bay Area. Photo taken in Mission Peak Regional Preserve, Fremont, CA. Summer 2014.

STRASSEN, from 30,000 feet

Volker Strassen
(Born in 1936, aged 80)



Original Strassen Paper (1969)

Numer. Math. 13, 354–356 (1969)

Gaussian Elimination is not Optimal

VOLKER STRASSEN*

Received December 12, 1968

1. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with less than $4.7 \cdot n^{\log 7}$ arithmetical operations (all logarithms in this paper are for base 2, thus $\log 7 \approx 2.8$; the usual method requires approximately $2n^3$ arithmetical operations). The algorithm induces algorithms for inverting a matrix of order n, solving a system of n linear equations in n unknowns, computing a determinant of order n etc. all requiring less than const $n^{\log 7}$ arithmetical operations.

This fact should be compared with the result of KLYUYEV and KOKOVKIN-SHCHERBAK [1] that Gaussian elimination for solving a system of linear equations is optimal if one restricts oneself to operations upon rows and columns as a whole. We also note that WINOGRAD [2] modifies the usual algorithms for matrix multiplication and inversion and for solving systems of linear equations, trading roughly half of the multiplications for additions and subtractions.

It is a pleasure to thank D. BRILLINGER for inspiring discussions about the present subject and ST. COOK and B. PARLETT for encouraging me to write this paper.

2. We define algorithms $\alpha_{m,k}$ which multiply matrices of order $m2^k$, by induction on k: $\alpha_{m,0}$ is the usual algorithm for matrix multiplication (requiring m^3 multiplications and $m^2(m-1)$ additions). $\alpha_{m,k}$ already being known, define $\alpha_{m,k+1}$ as follows:

If A, B are matrices of order $m2^{k+1}$ to be multiplied, write

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad AB = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

where the A_{ik}, B_{ik}, C_{ik} are matrices of order $m2^k$. Then compute

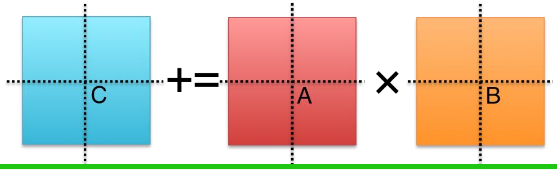
$$\begin{aligned}
\mathrm{I} &= (A_{11}+A_{22})(B_{11}+B_{22}), \\
\mathrm{II} &= (A_{21}+A_{22})B_{11}, \\
\mathrm{III} &= A_{11}(B_{12}-B_{22}), \\
\mathrm{IV} &= A_{22}(-B_{11}+B_{21}), \\
\mathrm{V} &= (A_{11}+A_{12})B_{22}, \\
\mathrm{VI} &= (-A_{11}+A_{21})(B_{11}+B_{12}), \\
\mathrm{VII} &= (A_{12}-A_{22})(B_{21}+B_{22}),
\end{aligned}$$

* The results have been found while the author was at the Department of Statistics of the University of California, Berkeley. The author wishes to thank the National Science Foundation for their support (NSF GP-7454).

One-level Strassen's Algorithm (In theory)

Assume m, n, and k are all even. A, B, and C are $m \times k$, $k \times n$, $m \times n$ matrices, respectively. Letting

$$C = \begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix}, A = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix}, B = \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}$$



We can compute $C := C + AB$ by

Direct Computation

$$\begin{aligned}
C_{00} &:= A_{00}B_{00} + A_{01}B_{10} + C_{00};\\
C_{01} &:= A_{00}B_{01} + A_{01}B_{11} + C_{01};\\
C_{10} &:= A_{10}B_{00} + A_{11}B_{10} + C_{10};\\
C_{11} &:= A_{10}B_{01} + A_{11}B_{11} + C_{11};
\end{aligned}$$

8 multiplications, 8 additions

Strassen's Algorithm

$$\begin{aligned}
M_0 &:= (A_{00} + A_{11})(B_{00} + B_{11});\\
M_1 &:= (A_{10} + A_{11})B_{00};\\
M_2 &:= A_{00}(B_{01} - B_{11});\\
M_3 &:= A_{11}(B_{10} - B_{00});\\
M_4 &:= (A_{00} + A_{01})B_{11};\\
M_5 &:= (A_{10} - A_{00})(B_{00} + B_{01});\\
M_6 &:= (A_{01} - A_{11})(B_{10} + B_{11});\\
C_{00} &:= M_0 + M_3 - M_4 + M_7 + C_{00};\\
C_{01} &:= M_2 + M_4 + C_{01};\\
C_{10} &:= M_1 + M_3 + C_{10};\\
C_{11} &:= M_0 - M_1 + M_2 + M_5 + C_{11}.
\end{aligned}$$

7 multiplications, 22 additions

*Strassen, Volker. "Gaussian elimination is not optimal." *Numerische Mathematik* 13, no. 4 (1969): 354-356.

Multi-level Strassen's Algorithm (In theory)

$M_0 := (A_{00}+A_{11})(B_{00}+B_{11});$
$M_1 := (A_{10}+A_{11})B_{00};$
$M_2 := A_{00}(B_{01}-B_{11});$
$M_3 := A_{11}(B_{10}-B_{00});$
$M_4 := (A_{00}+A_{01})B_{11};$
$M_5 := (A_{10}-A_{00})(B_{00}+B_{01});$
$M_6 := (A_{01}-A_{11})(B_{10}+B_{11});$
$C_{00} += M_0 + M_3 - M_4 + M_6$
$C_{01} += M_2 + M_4$
$C_{10} += M_1 + M_3$
$C_{11} += M_0 - M_1 + M_2 + M_5$

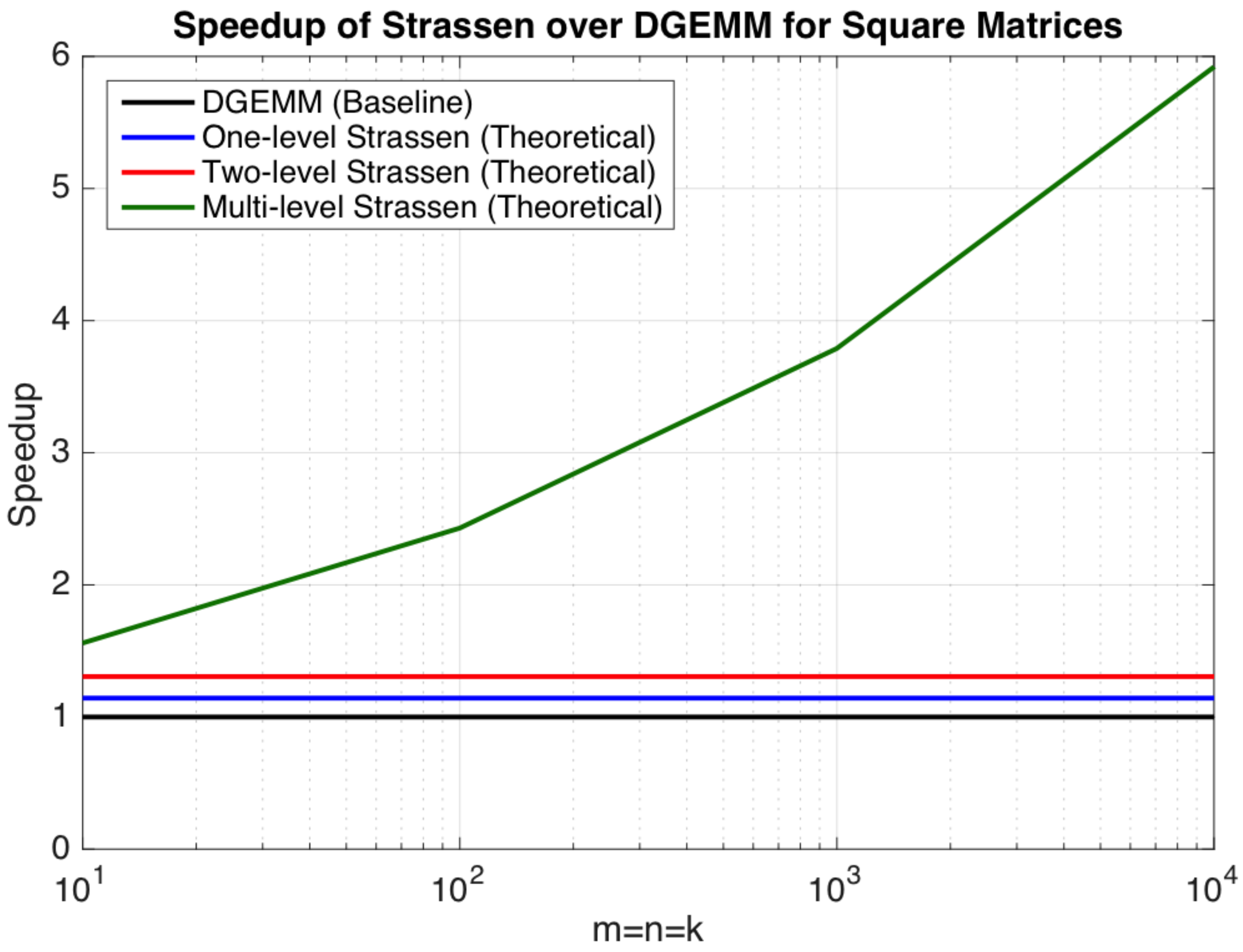
- One-level Strassen (1+14.3% speedup)
  - 8 multiplications → 7 multiplications ;
- Two-level Strassen (1+30.6% speedup)
  - 64 multiplications → 49 multiplications;
- *d*-level Strassen ($n^3/n^{2.803}$ speedup)
  - 8^d multiplications → 7^d multiplications;

If originally $m = n = k = 2^d$, where d is an integer, then the cost becomes
$(7/8)^{\log_2(n)} 2n^3 = n^{\log_2(7/8)} 2n^3 \approx 2n^{2.807}$ flops.

Multi-level Strassen's Algorithm (In theory)

$M_0 := (A_{00}+A_{11})(B_{00}+B_{11});$
$M_1 := (A_{10}+A_{11})B_{00};$
$M_2 := A_{00}(B_{01}-B_{11});$
$M_3 := A_{11}(B_{10}-B_{00});$
$M_4 := (A_{00}+A_{01})B_{11};$
$M_5 := (A_{10}-A_{00})(B_{00}+B_{01});$
$M_6 := (A_{01}-A_{11})(B_{10}+B_{11});$
$C_{00} += M_0 + M_3 - M_4 + M_6$
$C_{01} += M_2 + M_4$
$C_{10} += M_1 + M_3$
$C_{11} += M_0 - M_1 + M_2 + M_5$

- One-level Strassen (1+14.3% speedup)
  - 8 multiplications → 7 multiplications ;
- Two-level Strassen (1+30.6% speedup)
  - 64 multiplications → 49 multiplications;
- *d*-level Strassen ($n^3/n^{2.803}$ speedup)
  - 8^d multiplications → 7^d multiplications;



Strassen's Algorithm (In practice)

$M_0 := (A_{00}+A_{11})(B_{00}+B_{11});$

$M_1 := (\mathrm{A}_{10}+\mathrm{A}_{11})\mathrm{B}_{00};$

$M_2 := A_{00}(B_{01}-B_{11});$

$M_3 := A_{11}(B_{10}-B_{00});$

$M_4 := (\mathrm{A}_{00}+\mathrm{A}_{01})\mathrm{B}_{11};$

$M_5 := (A_{10}-A_{00})(B_{00}+B_{01});$

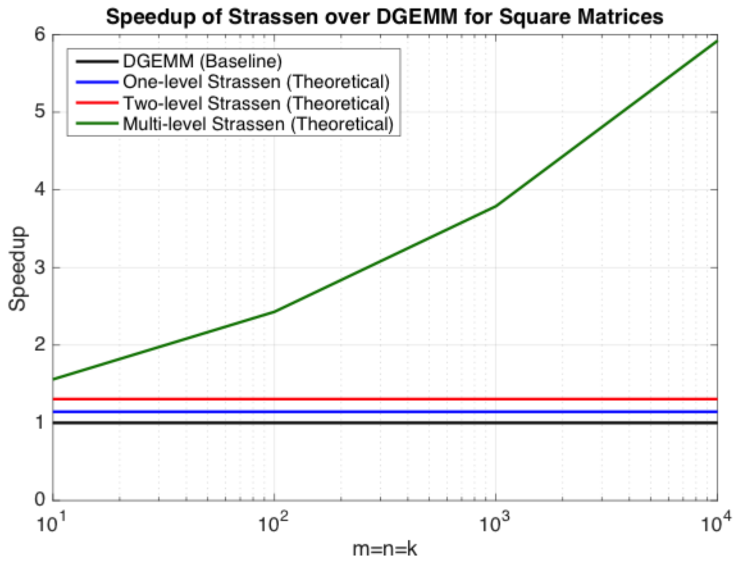
$M_6 := (A_{01}-A_{11})(B_{10}+B_{11});$

$C_{00} += M_0 + M_3 - M_4 + M_6$

$C_{01} += M_2 + M_4$

$C_{10} += M_1 + M_3$

$C_{11} += M_0 - M_1 + M_2 + M_5$



Strassen's Algorithm (In practice)

$M_0 := (\overbrace{A_{00}+A_{11}}^{T_0})(\overbrace{B_{00}+B_{11}}^{T_1});$

$M_1 := (\overbrace{A_{10}+A_{11}}^{T_2})B_{00};$

$M_2 := A_{00}(\overbrace{B_{01}-B_{11}}^{T_3});$

$M_3 := A_{11}(\overbrace{B_{10}-B_{00}}^{T_4});$

$M_4 := (\overbrace{A_{00}+A_{01}}^{T_5})B_{11};$

$M_5 := (\overbrace{A_{10}-A_{00}}^{T_6})(\overbrace{B_{00}+B_{01}}^{T_7});$

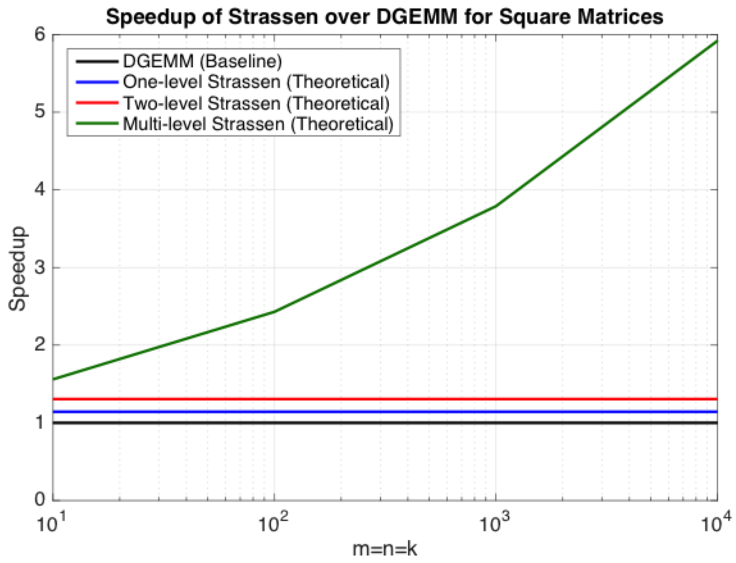
$M_6 := (\overbrace{A_{01}-A_{11}}^{T_8})(\overbrace{B_{10}+B_{11}}^{T_9});$

$C_{00} += M_0 + M_3 - M_4 + M_6$

$C_{01} += M_2 + M_4$

$C_{10} += M_1 + M_3$

$C_{11} += M_0 - M_1 + M_2 + M_5$



- One-level Strassen (1+14.3% speedup)
  - 7 multiplications + 22 additions;
- Two-level Strassen (1+30.6% speedup)
  - 49 multiplications + 344 additions;

Strassen's Algorithm (In practice)

$M_0 := (\overbrace{A_{00}+A_{11}}^{T_0})(\overbrace{B_{00}+B_{11}}^{T_1});$

$M_1 := (\overbrace{A_{10}+A_{11}}^{T_2})B_{00};$

$M_2 := A_{00}(\overbrace{B_{01}-B_{11}}^{T_3});$

$M_3 := A_{11}(\overbrace{B_{10}-B_{00}}^{T_4});$

$M_4 := (\overbrace{A_{00}+A_{01}}^{T_5})B_{11};$

$M_5 := (\overbrace{A_{10}-A_{00}}^{T_6})(\overbrace{B_{00}+B_{01}}^{T_7});$

$M_6 := (\overbrace{A_{01}-A_{11}}^{T_8})(\overbrace{B_{10}+B_{11}}^{T_9});$

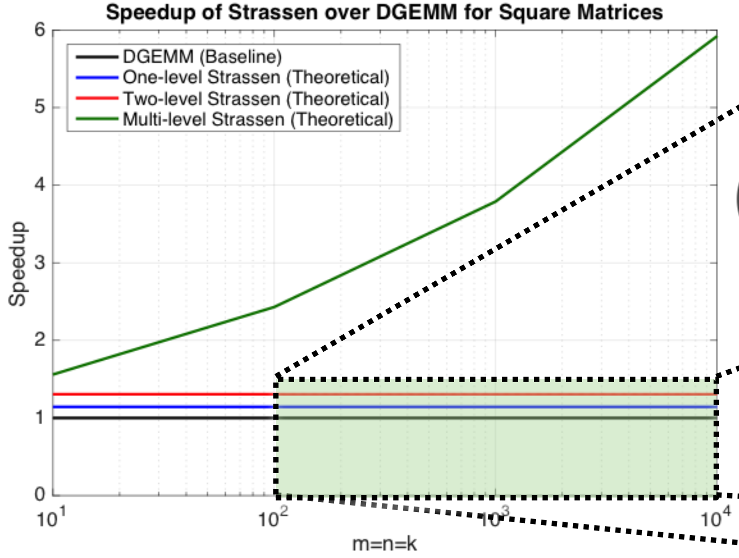
$C_{00} \mathrel{+}= M_0 + M_3 - M_4 + M_6$

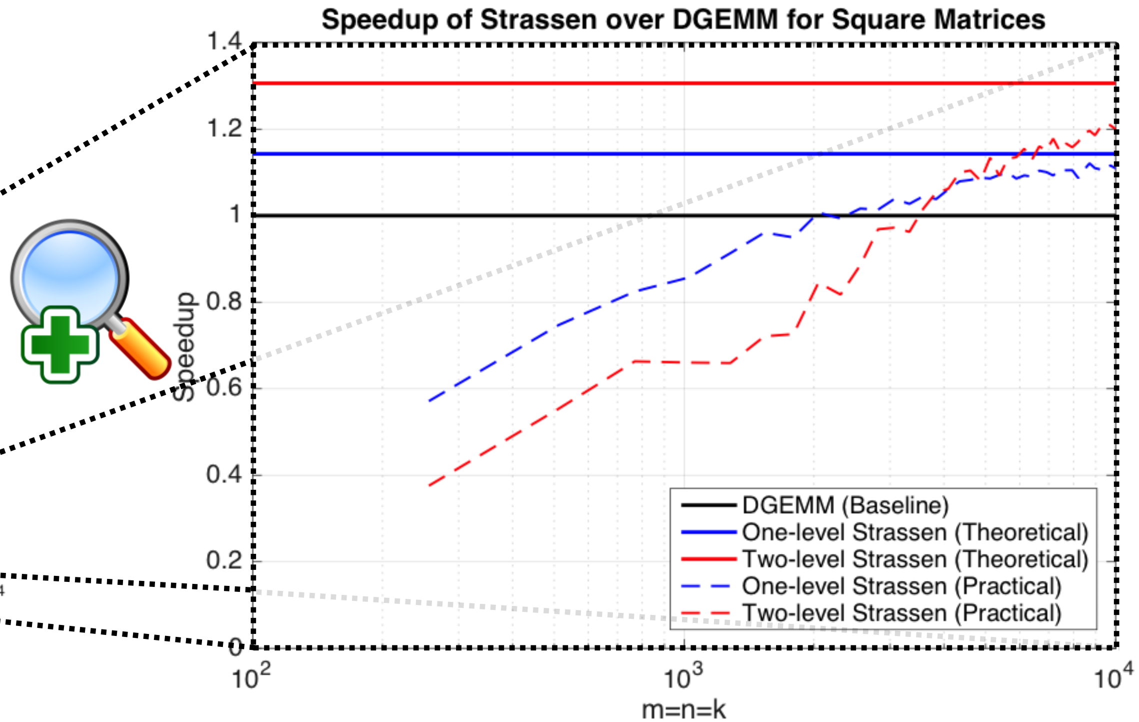
$C_{01} \mathrel{+}= M_2 + M_4$

$C_{10} \mathrel{+}= M_1 + M_3$

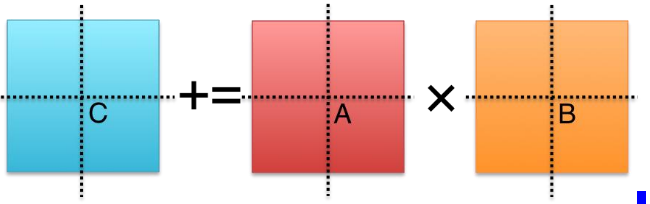
$C_{11} \mathrel{+}= M_0 - M_1 + M_2 + M_5$

- One-level Strassen (1+14.3% speedup)
  - ➢ 7 multiplications + 22 additions;
- Two-level Strassen (1+30.6% speedup)
  - ➢ 49 multiplications + 344 additions;
- *d*-level Strassen ($n^3/n^{2.803}$ speedup)
  - ➢ Numerical unstable; Not achievable



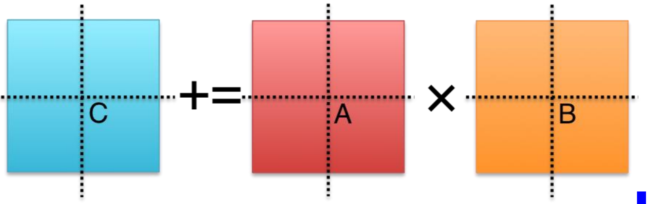


To achieve practical high performance of Strassen's algorithm……



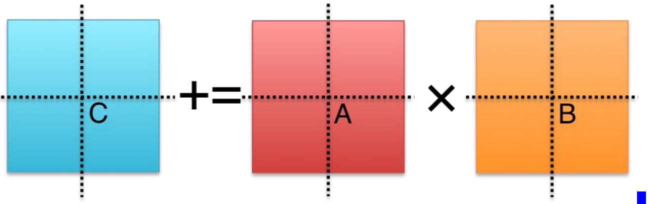
	Conventional Implementations	Our Implementations
Matrix Size	Must be large	
Matrix Shape	Must be square	
No Additional Workspace	✗	
Parallelism		

To achieve practical high performance of Strassen's algorithm……



	Conventional Implementations	Our Implementations
Matrix Size	Must be large	
Matrix Shape	Must be square	
No Additional Workspace	✗	
Parallelism	Usually task parallelism	

To achieve practical high performance of Strassen's algorithm……



	Conventional Implementations	Our Implementations
Matrix Size	Must be large	C += A × B
Matrix Shape	Must be square	C += A × B
No Additional Workspace	✗	✓
Parallelism	Usually task parallelism	Can be data parallelism

Outline

- Standard Matrix-matrix multiplication
- Strassen’s Algorithm Reloaded
- Theoretical Model and Analysis
- Performance Experiments
- Conclusion

Level-3 BLAS Matrix-Matrix Multiplication (GEMM)

- (General) matrix-matrix multiplication (GEMM) is supported in the level-3 BLAS* interface as

```
dgemm( transa, transb, m, n, k,
       alpha, A, lda, B, ldb,
       beta, C, ldc )
```

- Ignoring transa and transb, GEMM computes

$$C := \alpha AB + \beta C;$$

- We consider the simplified version of GEMM

$$C := \alpha AB + C$$

*Dongarra, Jack J., et al. "A set of level 3 basic linear algebra subprograms." *ACM Transactions on Mathematical Software (TOMS)* 16.1 (1990): 1-17.

State-of-the-art GEMM in BLIS

- BLAS-like Library Instantiation Software (BLIS) is a portable framework for instantiating BLAS-like dense linear algebra libraries.
  - ❑ Field Van Zee, and Robert van de Geijn. "BLIS: A Framework for Rapidly Instantiating BLAS Functionality." *ACM TOMS* 41.3 (2015): 14.
- BLIS provides a refactoring of GotoBLAS algorithm (best-known approach) to implement GEMM.
  - ❑ Kazushige Goto, and Robert van de Geijn. "High-performance implementation of the level-3 BLAS." *ACM TOMS* 35.1 (2008): 4.
  - ❑ Kazushige Goto, and Robert van de Geijn. "Anatomy of high-performance matrix multiplication." *ACM TOMS* 34.3 (2008): 12.
- GEMM implementation in BLIS has 6-layers of loops. The outer 5 loops are written in C. The inner-most loop (micro-kernel) is written in assembly for high performance.
  - ➢ Partition matrices into smaller blocks to fit into the different memory hierarchy.
  - ➢ The order of these loops is designed to utilize the cache reuse rate.

State-of-the-art GEMM in BLIS

- BLAS-like Library Instantiation Software (BLIS) is a portable framework for instantiating BLAS-like dense linear algebra libraries.

  ❑ Field Van Zee, and Robert van de Geijn. “BLIS: A Framework for Rapidly Instantiating BLAS Functionality." *ACM TOMS* 41.3 (2015): 14.

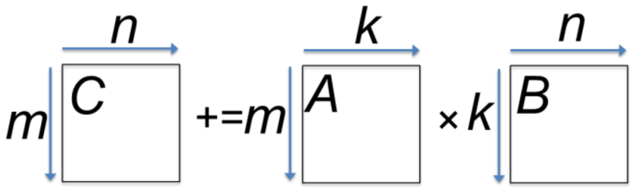
- BLIS provides a refactoring of GotoBLAS algorithm (best-known approach) to implement GEMM.

  ❑ Kazushige Goto, and Robert van de Geijn. "High-performance implementation of the level-3 BLAS." *ACM TOMS* 35.1 (2008): 4.
  ❑ Kazushige Goto, and Robert van de Geijn. "Anatomy of high-performance matrix multiplication." *ACM TOMS* 34.3 (2008): 12.

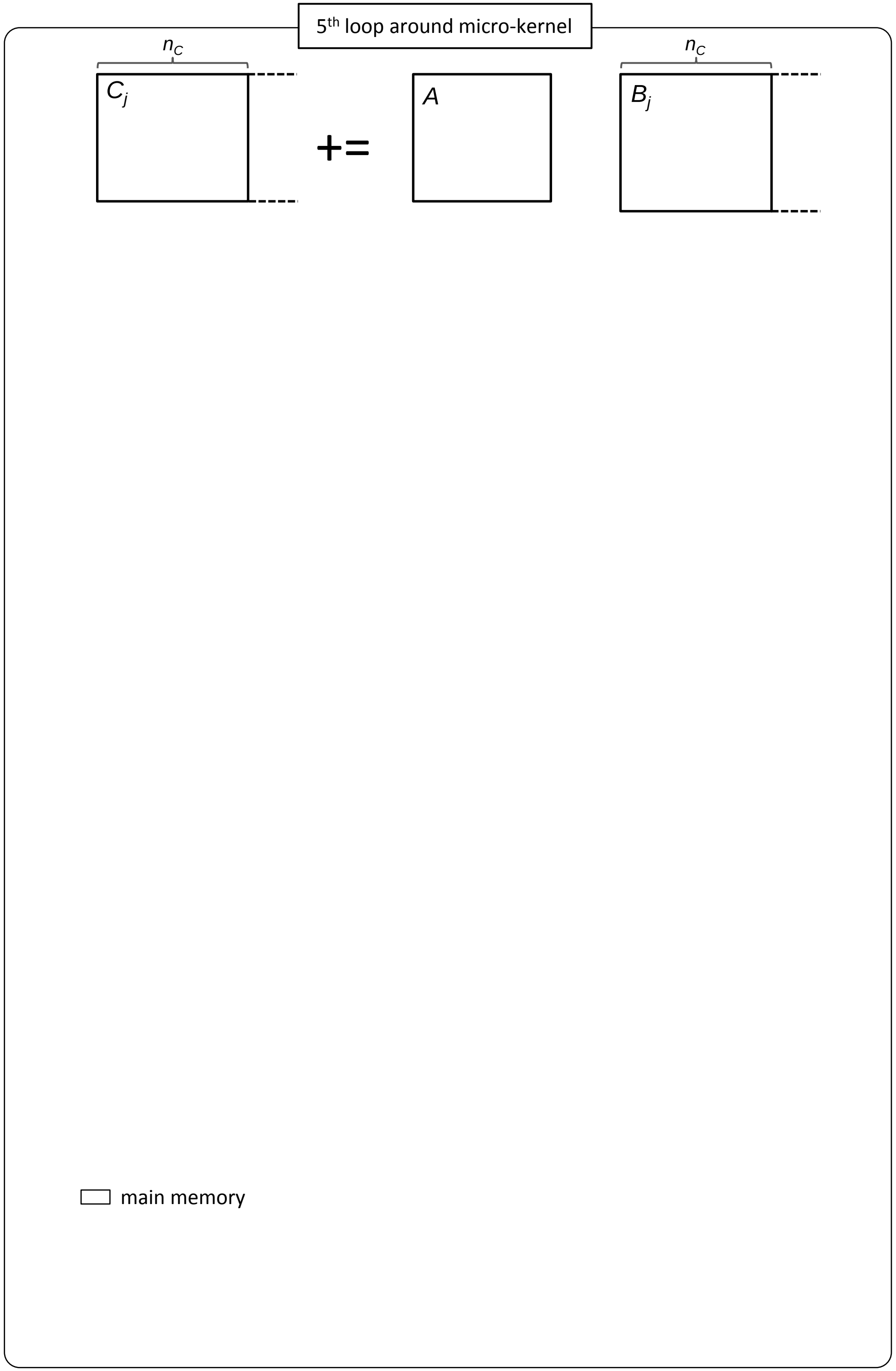
- GEMM implementation in BLIS has 6-layers of loops. The outer 5 loops are written in C. The inner-most loop (micro-kernel) is written in assembly for high performance.
  - ➢ Partition matrices into smaller blocks to fit into the different memory hierarchy.
  - ➢ The order of these loops is designed to utilize the cache reuse rate.
- BLIS opens the black box of GEMM, leading to many applications built on BLIS.

  ❑ Chenhan D. Yu, Jianyu Huang, Woody Austin, Bo Xiao, and George Biros. "Performance Optimization for the k-Nearest Neighbors Kernel on x86 Architectures." In *SC'15*.
  ❑ Jianyu Huang, Tyler Smith, Greg Henry, and Robert van de Geijn. “Strassen's Algorithm Reloaded.” In *SC'16*.
  ❑ Devin Matthews. “High-Performance Tensor Contraction without BLAS.”, arXiv:1607.00291
  ❑ Paul Springer, Paolo Bientinesi. “Design of a High-performance GEMM-like Tensor-Tensor Multiplication”, arXiv:1607.00145

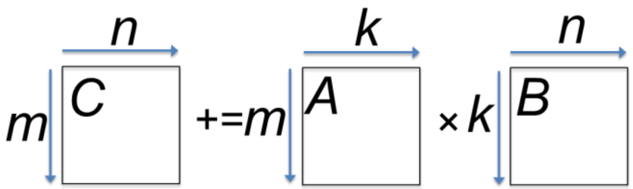
GotoBLAS algorithm for GEMM in BLIS



*Field G. Van Zee, and Tyler M. Smith. “Implementing high-performance complex matrix multiplication." In *ACM Transactions on Mathematical Software (TOMS), accepted pending modifications.*



GotoBLAS algorithm for GEMM in BLIS

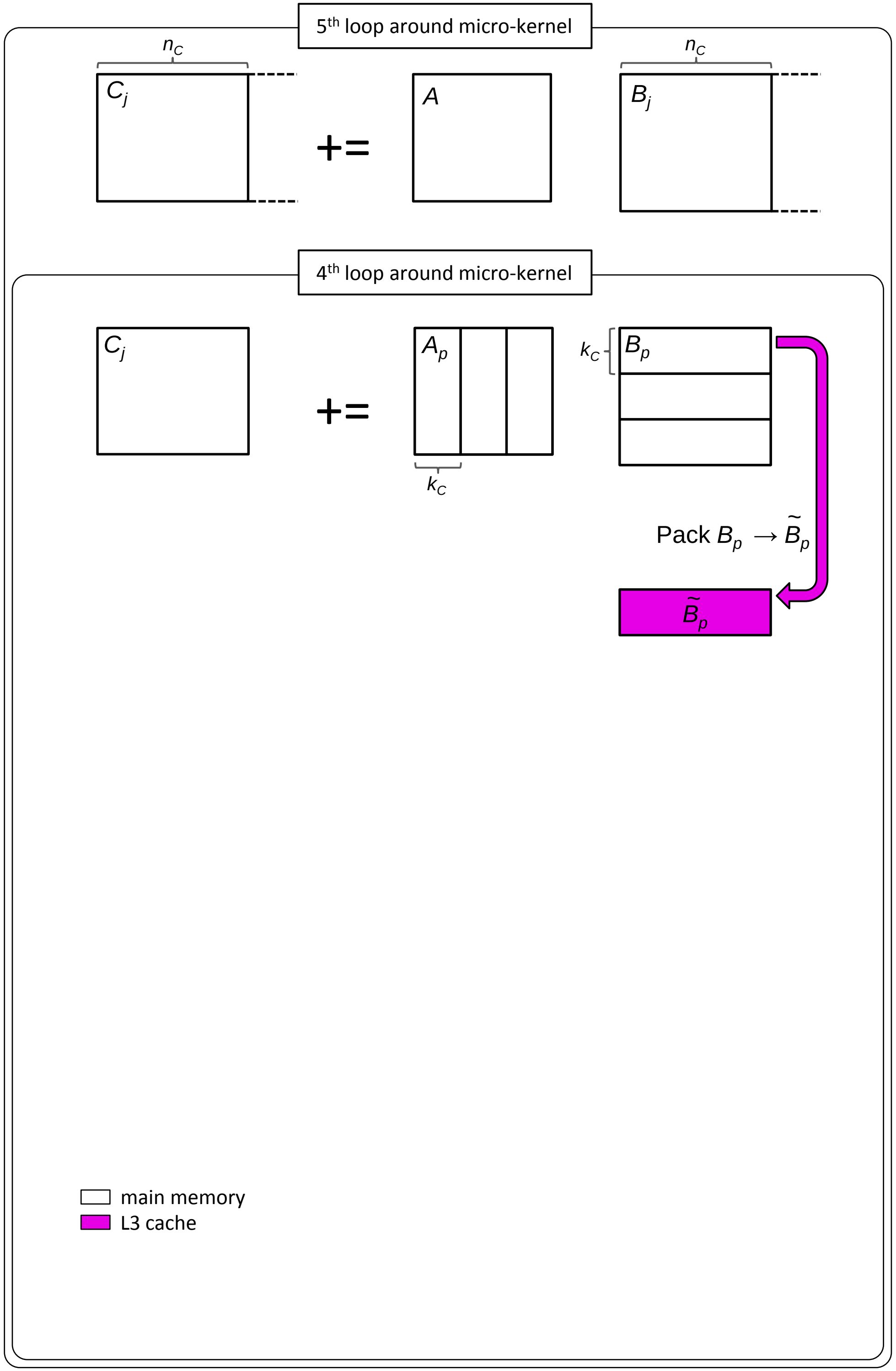


Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c

$\mathcal{J}_c = j_c : j_c + n_c - 1$

endfor

*Field G. Van Zee, and Tyler M. Smith. “Implementing high-performance complex matrix multiplication." In *ACM Transactions on Mathematical Software (TOMS), accepted pending modifications.*



GotoBLAS algorithm for GEMM in BLIS

$$C_{m \times n} \mathrel{+}= A_{m \times k} \times B_{k \times n}$$

Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c

$\mathcal{J}_c = j_c : j_c + n_c - 1$

Loop 4 **for** $p_c = 0 : k-1$ **steps of** k_c

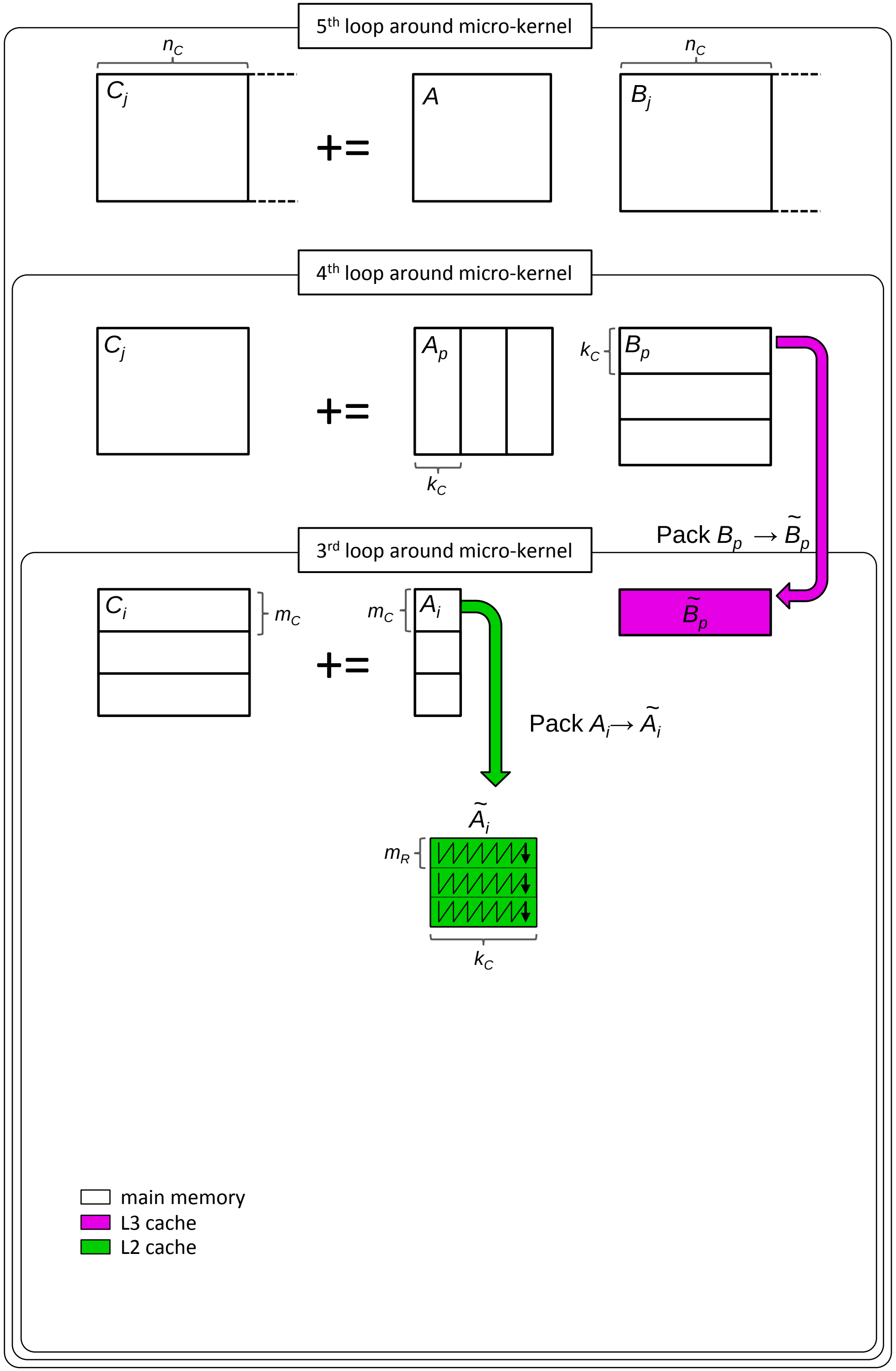
$\mathcal{P}_c = p_c : p_c + k_c - 1$

$B(\mathcal{P}_c, \mathcal{J}_c) \to \widetilde{B}_p$

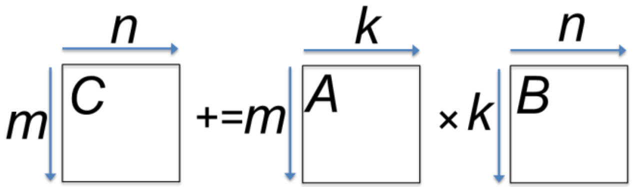
endfor

endfor

*Field G. Van Zee, and Tyler M. Smith. "Implementing high-performance complex matrix multiplication." In *ACM Transactions on Mathematical Software (TOMS), accepted pending modifications.*



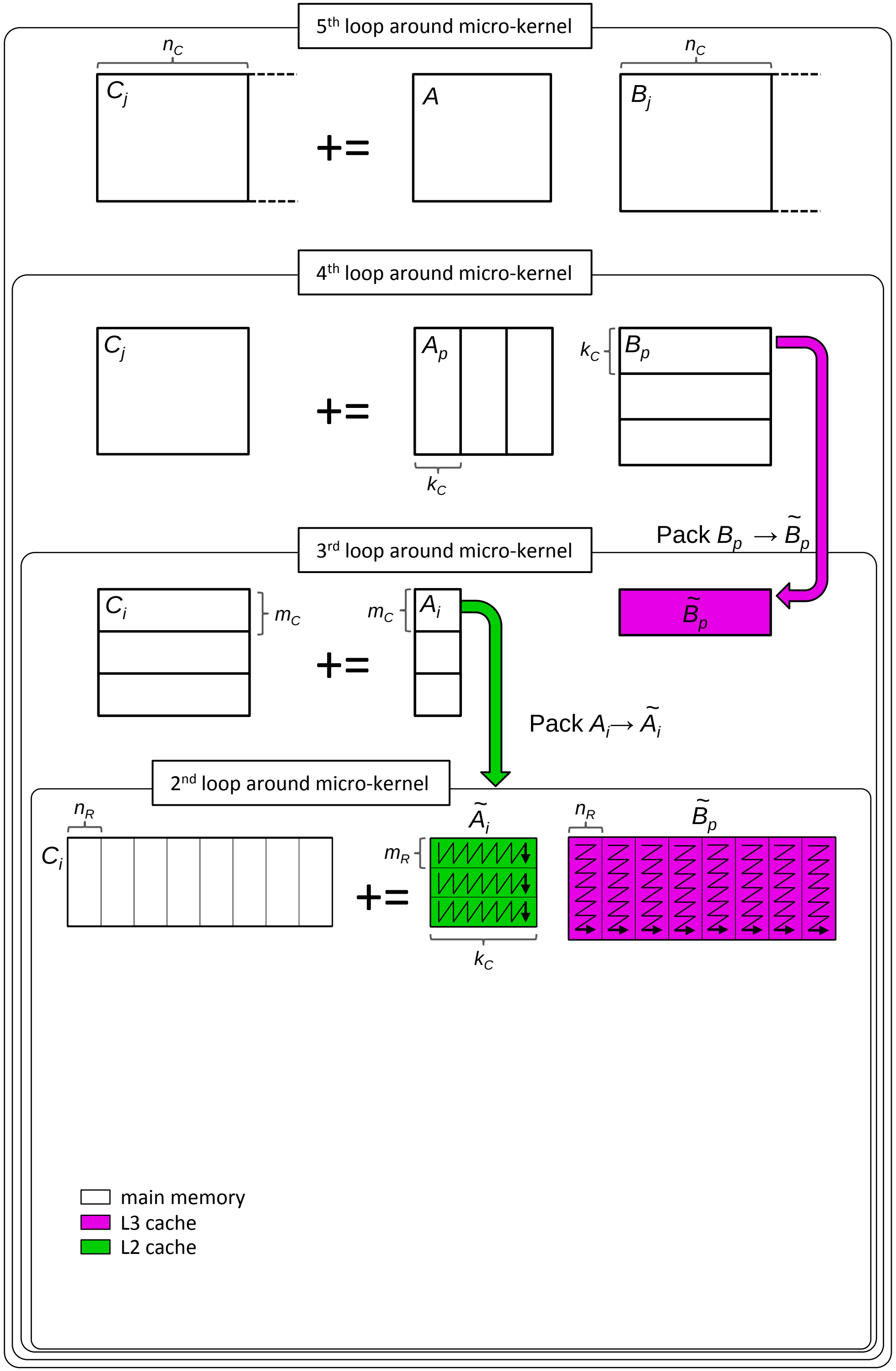
GotoBLAS algorithm for GEMM in BLIS



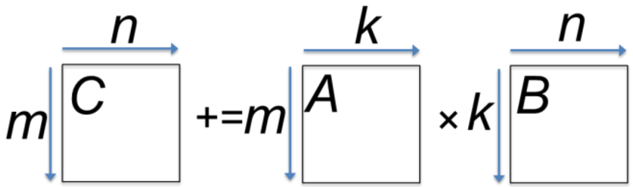
Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c
$\mathcal{J}_c = j_c : j_c + n_c - 1$
Loop 4 **for** $p_c = 0 : k-1$ **steps of** k_c
$\mathcal{P}_c = p_c : p_c + k_c - 1$
$B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \widetilde{B}_p$
Loop 3 **for** $i_c = 0 : m-1$ **steps of** m_c
$\mathcal{I}_c = i_c : i_c + m_c - 1$
$A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \widetilde{A}_i$

endfor
endfor
endfor

*Field G. Van Zee, and Tyler M. Smith. "Implementing high-performance complex matrix multiplication." In *ACM Transactions on Mathematical Software (TOMS), accepted pending modifications.*



GotoBLAS algorithm for GEMM in BLIS

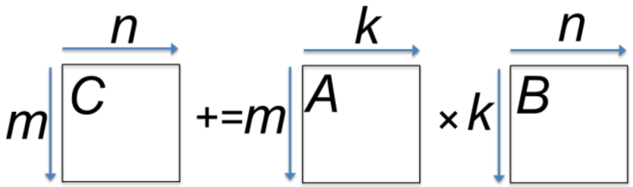


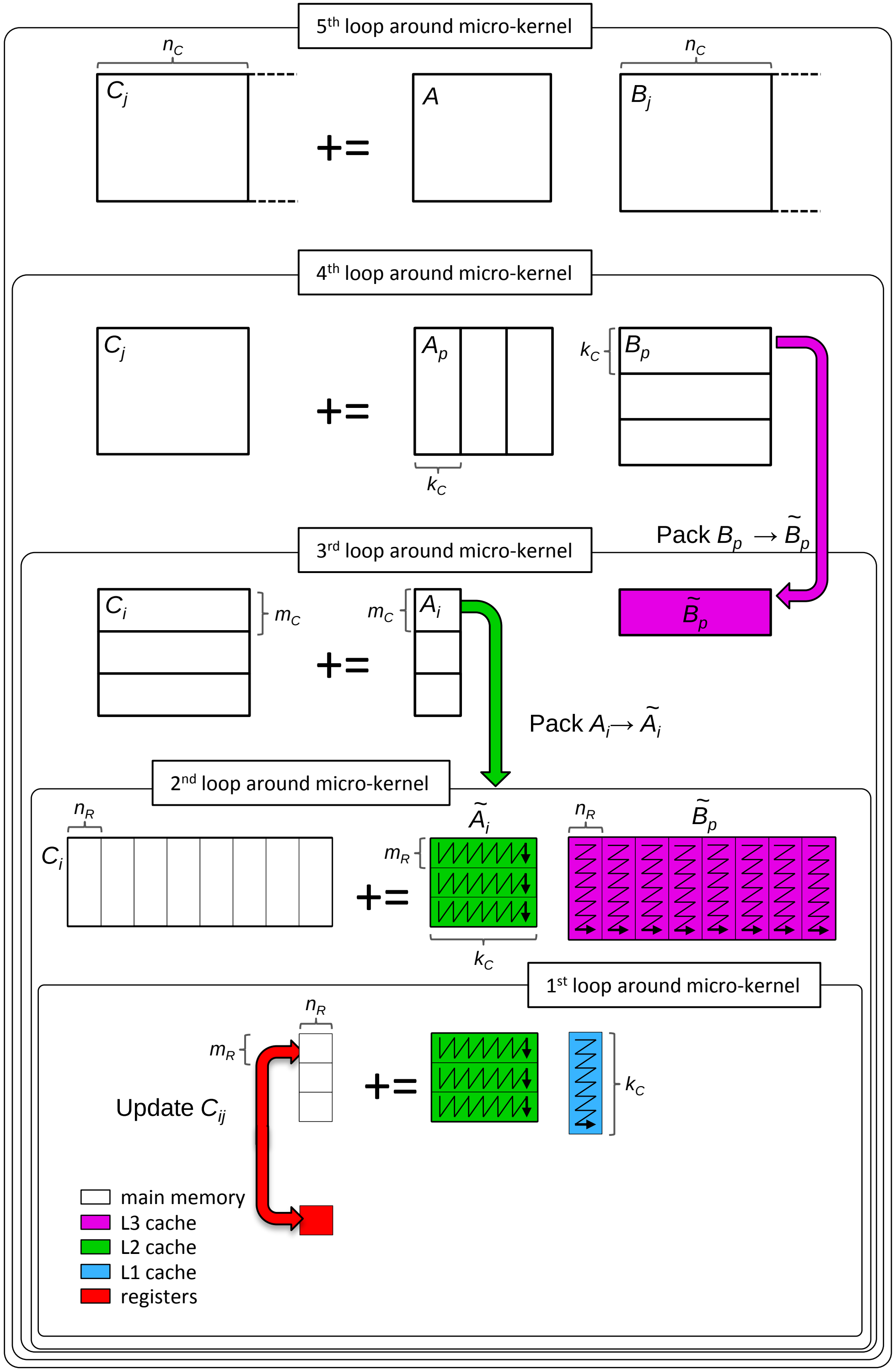
Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c
$\mathcal{J}_c = j_c : j_c + n_c - 1$
Loop 4 **for** $p_c = 0 : k-1$ **steps of** k_c
$\mathcal{P}_c = p_c : p_c + k_c - 1$
$B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \tilde{B}_p$
Loop 3 **for** $i_c = 0 : m-1$ **steps of** m_c
$\mathcal{I}_c = i_c : i_c + m_c - 1$
$A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \tilde{A}_i$
`// macro-kernel`
Loop 2 **for** $j_r = 0 : n_c - 1$ **steps of** n_r
$\mathcal{J}_r = j_r : j_r + n_r - 1$

endfor
endfor
endfor
endfor

*Field G. Van Zee, and Tyler M. Smith. "Implementing high-performance complex matrix multiplication." In *ACM Transactions on Mathematical Software (TOMS), accepted pending modifications.*

GotoBLAS algorithm for GEMM in BLIS





Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c

$\mathcal{J}_c = j_c : j_c + n_c - 1$

Loop 4 **for** $p_c = 0 : k-1$ **steps of** k_c

$\mathcal{P}_c = p_c : p_c + k_c - 1$

$B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \tilde{B}_p$

Loop 3 **for** $i_c = 0 : m-1$ **steps of** m_c

$\mathcal{I}_c = i_c : i_c + m_c - 1$

$A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \tilde{A}_i$

```
// macro-kernel
```

Loop 2 **for** $j_r = 0 : n_c - 1$ **steps of** n_r

$\mathcal{J}_r = j_r : j_r + n_r - 1$

Loop 1 **for** $i_r = 0 : m_c - 1$ **steps of** m_r

$\mathcal{I}_r = i_r : i_r + m_r - 1$

endfor

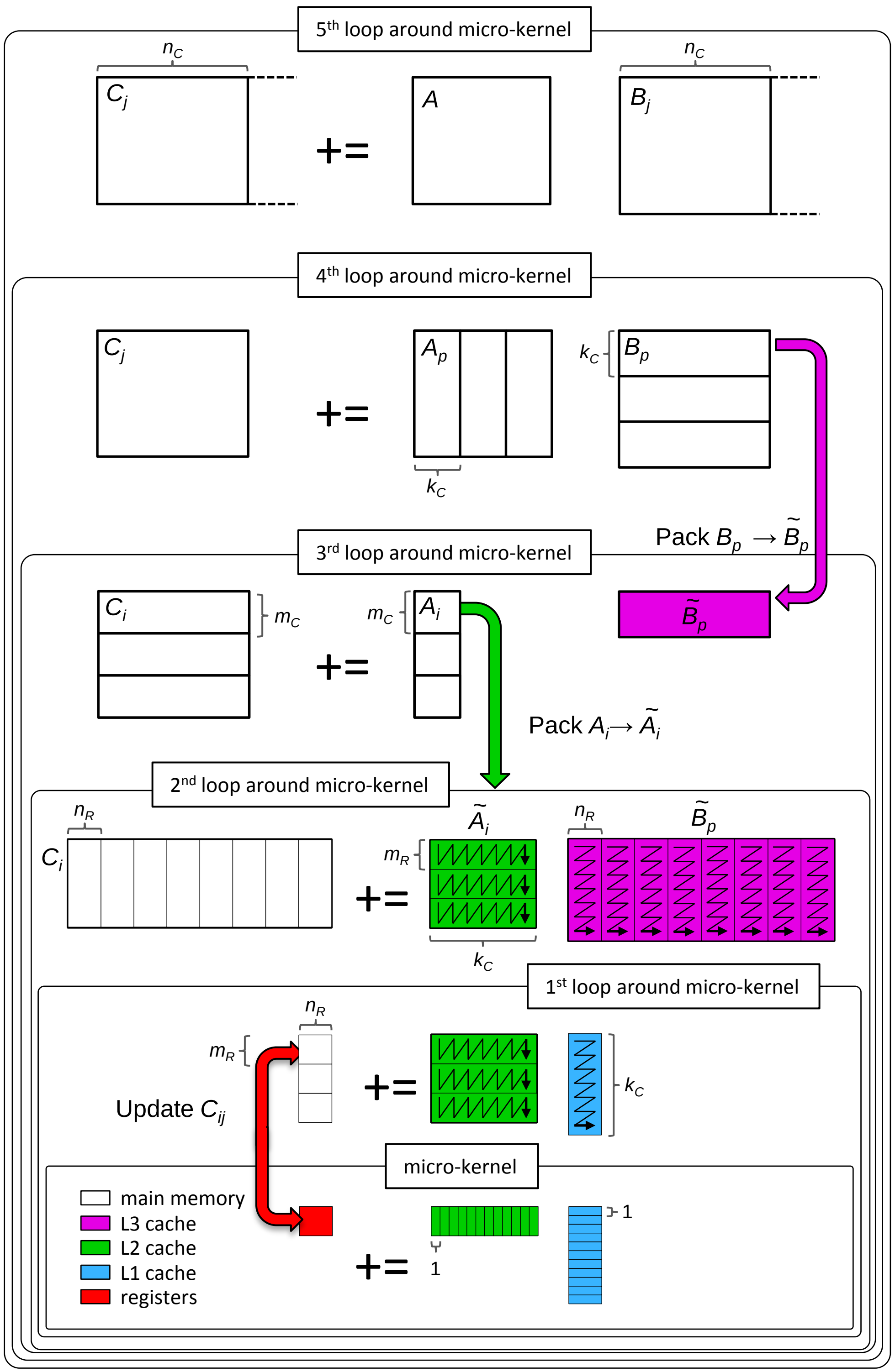
endfor

endfor

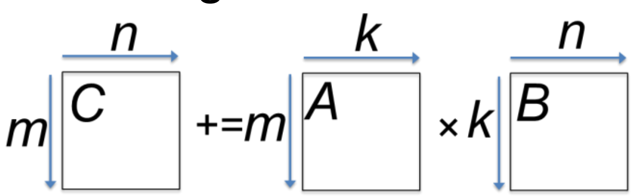
endfor

endfor

*Field G. Van Zee, and Tyler M. Smith. "Implementing high-performance complex matrix multiplication." In *ACM Transactions on Mathematical Software (TOMS), accepted pending modifications.*



GotoBLAS algorithm for GEMM in BLIS



Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c

$\mathcal{J}_c = j_c : j_c + n_c - 1$

Loop 4 **for** $p_c = 0 : k-1$ **steps of** k_c

$\mathcal{P}_c = p_c : p_c + k_c - 1$

$B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \tilde{B}_p$

Loop 3 **for** $i_c = 0 : m-1$ **steps of** m_c

$\mathcal{I}_c = i_c : i_c + m_c - 1$

$A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \tilde{A}_i$

`// macro-kernel`

Loop 2 **for** $j_r = 0 : n_c - 1$ **steps of** n_r

$\mathcal{J}_r = j_r : j_r + n_r - 1$

Loop 1 **for** $i_r = 0 : m_c - 1$ **steps of** m_r

$\mathcal{I}_r = i_r : i_r + m_r - 1$

`//micro-kernel`

Loop 0 **for** $p_r = 0 : p_c - 1$ **steps of** 1

$C_c(\mathcal{I}_r, \mathcal{J}_r) \mathrel{+}= \alpha \tilde{A}_i(\mathcal{I}_r, p_r)\, \tilde{B}_p(p_r, \mathcal{J}_r)$

endfor

endfor

endfor

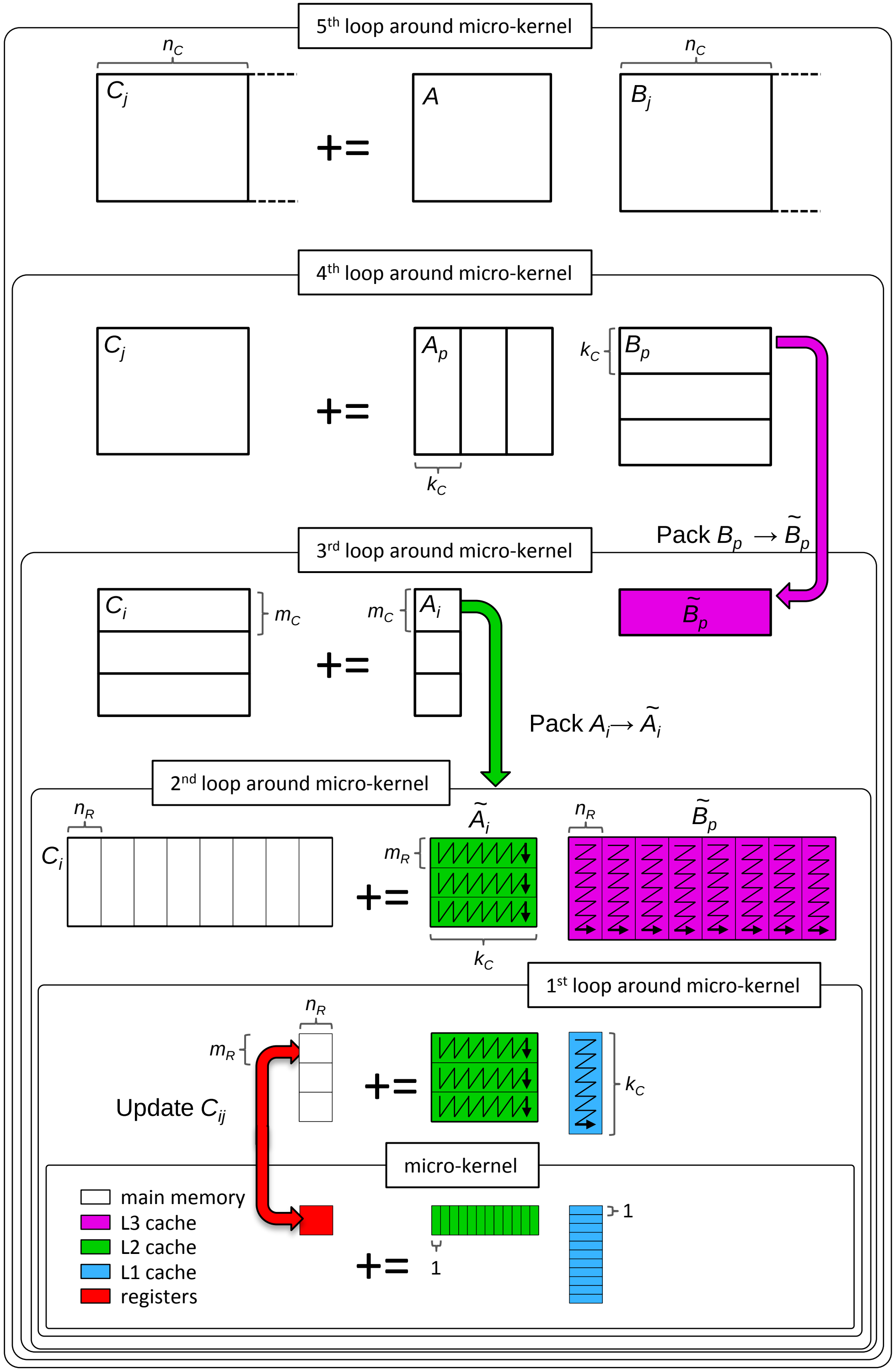
endfor

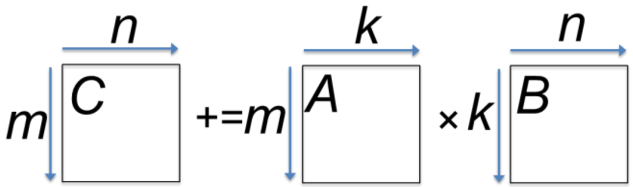
endfor

endfor

*Field G. Van Zee, and Tyler M. Smith. "Implementing high-performance complex matrix multiplication." In *ACM Transactions on Mathematical Software (TOMS), accepted pending modifications.*

GotoBLAS algorithm for GEMM in BLIS





Loop 5 for $j_c = 0 : n-1$ steps of n_c

$\mathcal{J}_c = j_c : j_c + n_c - 1$

Loop 4 for $p_c = 0 : k-1$ steps of k_c

$\mathcal{P}_c = p_c : p_c + k_c - 1$

$B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \tilde{B}_p$

Loop 3 for $i_c = 0 : m-1$ steps of m_c

$\mathcal{I}_c = i_c : i_c + m_c - 1$

$A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \tilde{A}_i$

`// macro-kernel`

Loop 2 for $j_r = 0 : n_c - 1$ steps of n_r

$\mathcal{J}_r = j_r : j_r + n_r - 1$

Loop 1 for $i_r = 0 : m_c - 1$ steps of m_r

$\mathcal{I}_r = i_r : i_r + m_r - 1$

`//micro-kernel`

Loop 0 for $p_r = 0 : p_c - 1$ steps of 1

$C_c(\mathcal{I}_r, \mathcal{J}_r) \mathrel{+}= \alpha \tilde{A}_i(\mathcal{I}_r, p_r)\, \tilde{B}_p(p_r, \mathcal{J}_r)$

endfor

endfor

endfor

endfor

endfor

endfor

*Field G. Van Zee, and Tyler M. Smith. "Implementing high-performance complex matrix multiplication." In *ACM Transactions on Mathematical Software (TOMS), accepted pending modifications.*

Outline

- Standard Matrix-matrix multiplication
- Strassen's Algorithm Reloaded
- Theoretical Model and Analysis
- Performance Experiments
- Conclusion

One-level Strassen's Algorithm Reloaded

$M_0 := \alpha(A_{00}+A_{11})(B_{00}+B_{11});$
$M_1 := \alpha(A_{10}+A_{11})B_{00};$
$M_2 := \alpha A_{00}(B_{01}-B_{11});$
$M_3 := \alpha A_{11}(B_{10}-B_{00});$
$M_4 := \alpha(A_{00}+A_{01})B_{11};$
$M_5 := \alpha(A_{10}-A_{00})(B_{00}+B_{01});$
$M_6 := \alpha(A_{01}-A_{11})(B_{10}+B_{11});$
$C_{00} \mathrel{+}= M_0 + M_3 - M_4 + M_6$
$C_{01} \mathrel{+}= M_2 + M_4$
$C_{10} \mathrel{+}= M_1 + M_3$
$C_{11} \mathrel{+}= M_0 - M_1 + M_2 + M_5$

$\boxed{M_0 := \alpha(A_{00}+A_{11})(B_{00}+B_{11}); \quad C_{00} \mathrel{+}= M_0;\ C_{11} \mathrel{+}= M_0;}$
$M_1 := \alpha(A_{10}+A_{11})B_{00}; \quad C_{10} \mathrel{+}= M_1;\ C_{11} \mathrel{-}= M_1;$
$M_2 := \alpha A_{00}(B_{01}-B_{11}); \quad C_{01} \mathrel{+}= M_2;\ C_{11} \mathrel{+}= M_2;$
$M_3 := \alpha A_{11}(B_{10}-B_{00}); \quad C_{00} \mathrel{+}= M_3;\ C_{10} \mathrel{+}= M_3;$
$M_4 := \alpha(A_{00}+A_{01})B_{11}; \quad C_{01} \mathrel{+}= M_4;\ C_{00} \mathrel{-}= M_4;$
$M_5 := \alpha(A_{10}-A_{00})(B_{00}+B_{01}); \quad C_{11} \mathrel{+}= M_5;$
$M_6 := \alpha(A_{01}-A_{11})(B_{10}+B_{11}); \quad C_{00} \mathrel{+}= M_6;$

$\boxed{M := \alpha(X+Y)(V+W); \quad C \mathrel{+}= M; \quad D \mathrel{+}= M;}$

General operation for one-level Strassen:
$\boxed{M := \alpha(X+\delta Y)(V+\varepsilon W); \quad C \mathrel{+}= \gamma_0 M;\ D \mathrel{+}= \gamma_1 M; \quad \gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}.}$

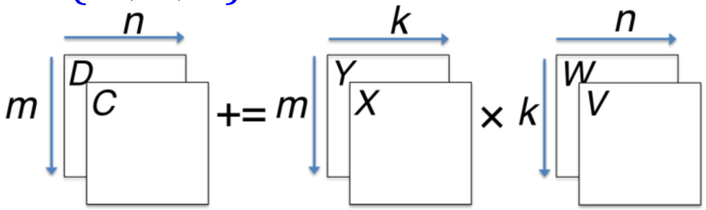
High-performance implementation of the general operation?

$M := \alpha(X+\delta Y)(V+\varepsilon W);$ $C += \gamma_0 M;$ $D += \gamma_1 M;$

$\gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}.$

$M := \alpha(X+\delta Y)(V+\varepsilon W)$; $C += \gamma_0 M$; $D += \gamma_1 M$;

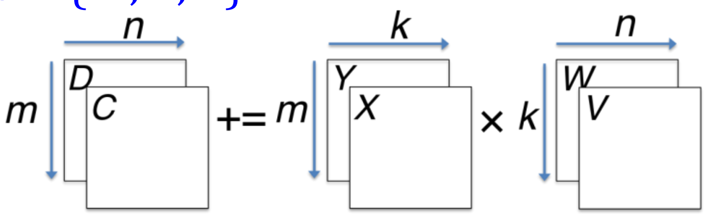
$\gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}$.



***Jianyu Huang**, Tyler Smith, Greg Henry, and Robert van de Geijn. "Strassen's Algorithm Reloaded." In *SC'16*.

$M := \alpha(X+\delta Y)(V+\varepsilon W); \qquad C \mathrel{+}= \gamma_0 M;\ D \mathrel{+}= \gamma_1 M;$

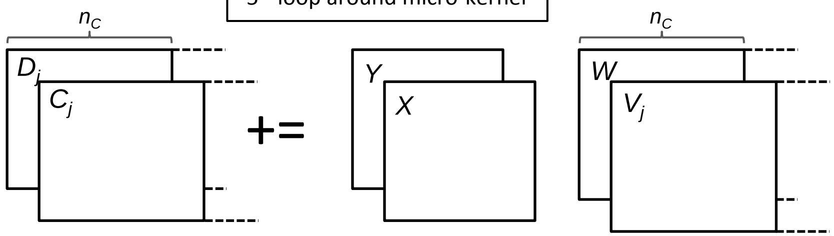
$\gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}.$



Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c

$\mathcal{J}_c = j_c : j_c + n_c - 1$

5th loop around micro-kernel



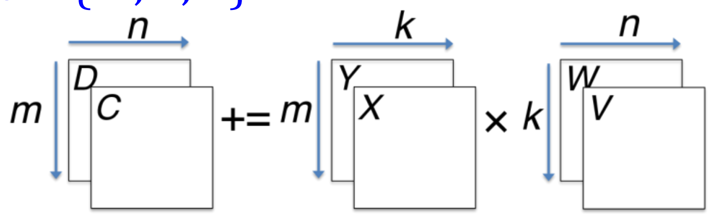
main memory

endfor

*__Jianyu Huang__, Tyler Smith, Greg Henry, and Robert van de Geijn. "Strassen's Algorithm Reloaded." In *SC'16*.

$M := \alpha(X+\delta Y)(V+\varepsilon W)$; $C += \gamma_0 M$; $D += \gamma_1 M$;

$\gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}$.



Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c

$\mathcal{J}_c = j_c : j_c + n_c - 1$

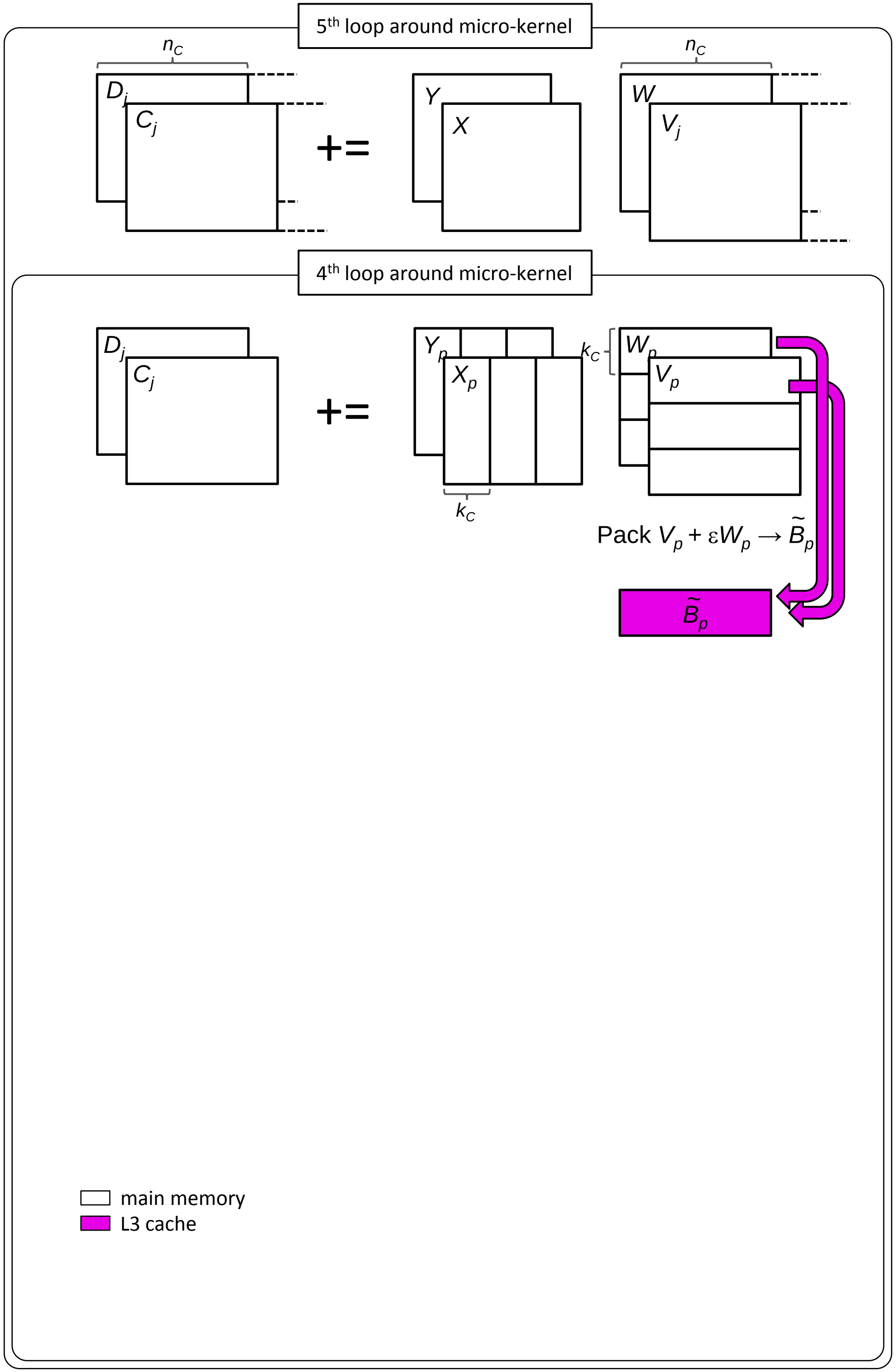
Loop 4 **for** $p_c = 0 : k-1$ **steps of** k_c

$\mathcal{P}_c = p_c : p_c + k_c - 1$

$V(\mathcal{P}_c, \mathcal{J}_c) + \epsilon W(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \widetilde{B}_p$

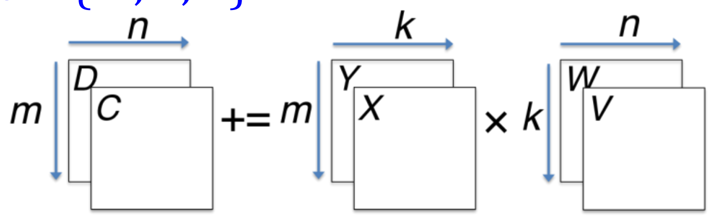
endfor

endfor



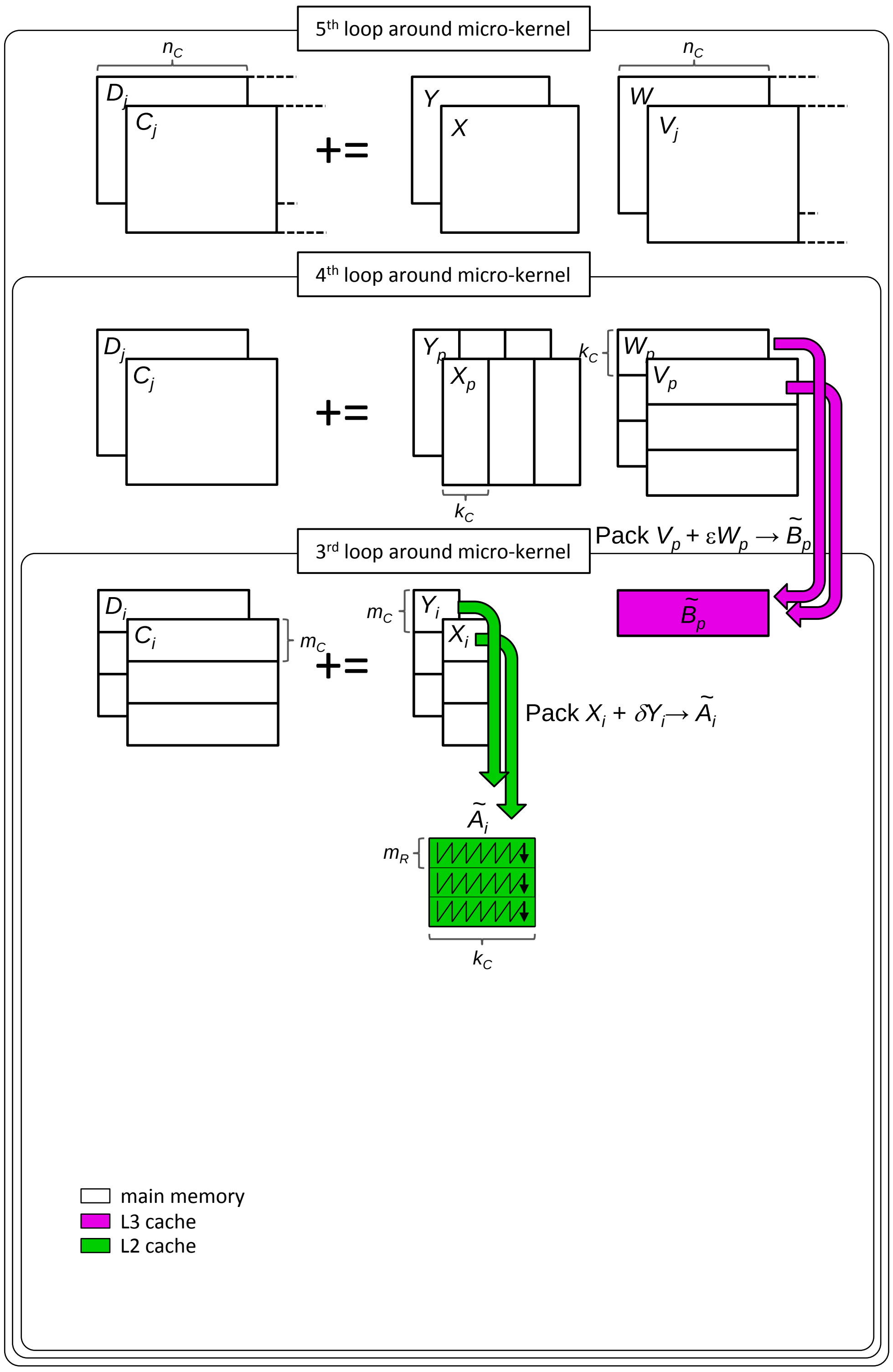
*__Jianyu Huang__, Tyler Smith, Greg Henry, and Robert van de Geijn. "Strassen's Algorithm Reloaded." In *SC'16*.

$M := \alpha(X+\delta Y)(V+\varepsilon W)$; $C += \gamma_0 M$; $D += \gamma_1 M$;
$\gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}$.



Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c
$\mathcal{J}_c = j_c : j_c + n_c - 1$
Loop 4 **for** $p_c = 0 : k-1$ **steps of** k_c
$\mathcal{P}_c = p_c : p_c + k_c - 1$
$V(\mathcal{P}_c, \mathcal{J}_c) + \epsilon W(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \widetilde{B}_p$
Loop 3 **for** $i_c = 0 : m-1$ **steps of** m_c
$\mathcal{I}_c = i_c : i_c + m_c - 1$
$X(\mathcal{I}_c, \mathcal{P}_c) + \delta Y(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \widetilde{A}_i$

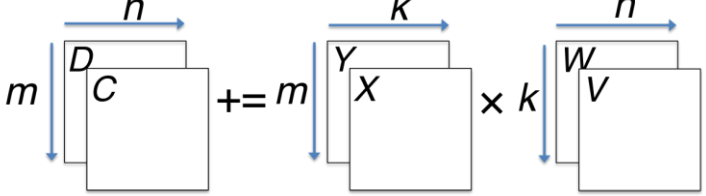
endfor
endfor
endfor



*__Jianyu Huang__, Tyler Smith, Greg Henry, and Robert van de Geijn. "Strassen's Algorithm Reloaded." In *SC'16*.

$M := \alpha(X+\delta Y)(V+\varepsilon W);\quad C \mathrel{+}= \gamma_0 M;\ D \mathrel{+}= \gamma_1 M;$

$\gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}.$



Loop 5 $\quad$ **for** $j_c = 0 : n-1$ **steps of** n_c

$\qquad \mathcal{J}_c = j_c : j_c + n_c - 1$

Loop 4 $\quad$ **for** $p_c = 0 : k-1$ **steps of** k_c

$\qquad \mathcal{P}_c = p_c : p_c + k_c - 1$

$\qquad V(\mathcal{P}_c, \mathcal{J}_c) + \epsilon W(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \widetilde{B}_p$

Loop 3 $\quad$ **for** $i_c = 0 : m-1$ **steps of** m_c

$\qquad \mathcal{I}_c = i_c : i_c + m_c - 1$

$\qquad X(\mathcal{I}_c, \mathcal{P}_c) + \delta Y(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \widetilde{A}_i$

`// macro-kernel`

Loop 2 $\quad$ **for** $j_r = 0 : n_c - 1$ **steps of** n_r

$\qquad \mathcal{J}_r = j_r : j_r + n_r - 1$

Loop 1 $\quad$ **for** $i_r = 0 : m_c - 1$ **steps of** m_r

$\qquad \mathcal{I}_r = i_r : i_r + m_r - 1$

`//micro-kernel`

Loop 0 $\quad$ **for** $p_r = 0 : p_c - 1$ **steps of** 1

$\qquad M_r(\mathcal{I}_r, \mathcal{J}_r) \mathrel{+}= \widetilde{A}_i(\mathcal{I}_r, p_r)\, \widetilde{B}_p(p_r, \mathcal{J}_r)$

endfor

$C(\mathcal{I}_r + i_c, \mathcal{J}_r + j_c) \mathrel{+}= \alpha\gamma_0 M_r(\mathcal{I}_r, \mathcal{J}_r)$

$D(\mathcal{I}_r + i_c, \mathcal{J}_r + j_c) \mathrel{+}= \alpha\gamma_1 M_r(\mathcal{I}_r, \mathcal{J}_r)$

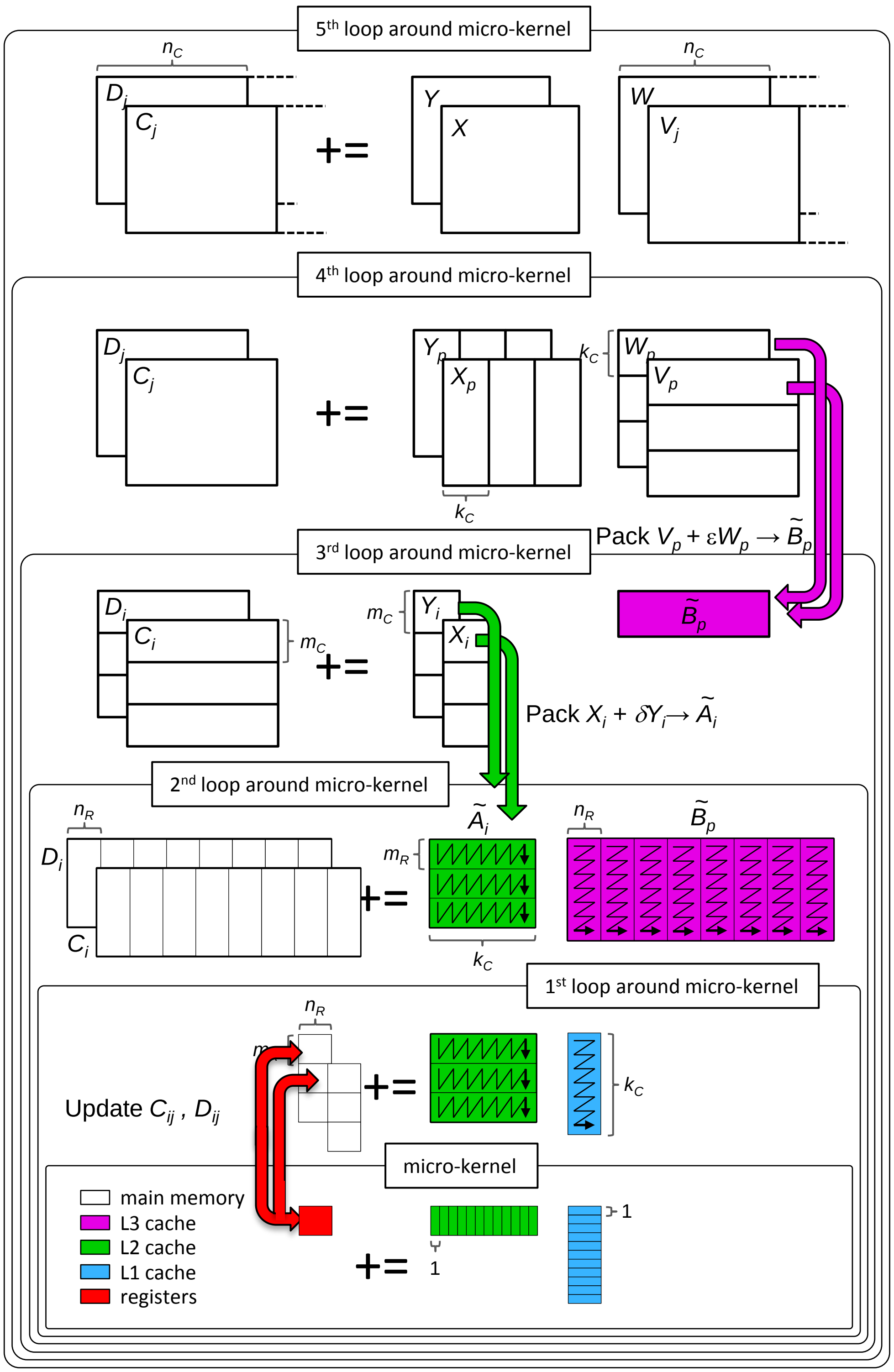
endfor

endfor

endfor

endfor

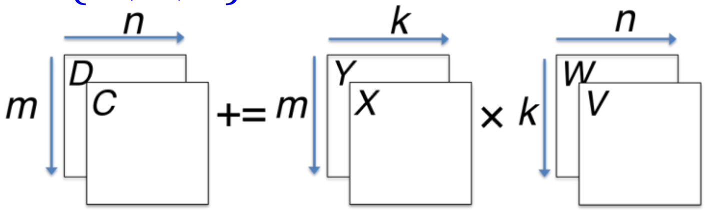
endfor



*__Jianyu Huang__, Tyler Smith, Greg Henry, and Robert van de Geijn. “Strassen’s Algorithm Reloaded.” In *SC’16*.

$M := \alpha(X+\delta Y)(V+\varepsilon W);\qquad C += \gamma_0 M;\ D += \gamma_1 M;$

$\gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}.$



Loop 5 for $j_c = 0 : n-1$ steps of n_c

$\mathcal{J}_c = j_c : j_c + n_c - 1$

Loop 4 for $p_c = 0 : k-1$ steps of k_c

$\mathcal{P}_c = p_c : p_c + k_c - 1$

$V(\mathcal{P}_c, \mathcal{J}_c) + \epsilon W(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \widetilde{B}_p$

Loop 3 for $i_c = 0 : m-1$ steps of m_c

$\mathcal{I}_c = i_c : i_c + m_c - 1$

$X(\mathcal{I}_c, \mathcal{P}_c) + \delta Y(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \widetilde{A}_i$

// macro-kernel

Loop 2 for $j_r = 0 : n_c - 1$ steps of n_r

$\mathcal{J}_r = j_r : j_r + n_r - 1$

Loop 1 for $i_r = 0 : m_c - 1$ steps of m_r

$\mathcal{I}_r = i_r : i_r + m_r - 1$

//micro-kernel

Loop 0 for $p_r = 0 : p_c - 1$ steps of 1

$M_r(\mathcal{I}_r, \mathcal{J}_r) += \widetilde{A}_i(\mathcal{I}_r, p_r)\ \widetilde{B}_p(p_r, \mathcal{J}_r)$

endfor

$C(\mathcal{I}_r + i_c, \mathcal{J}_r + j_c) += \alpha\gamma_0 M_r(\mathcal{I}_r, \mathcal{J}_r)$

$D(\mathcal{I}_r + i_c, \mathcal{J}_r + j_c) += \alpha\gamma_1 M_r(\mathcal{I}_r, \mathcal{J}_r)$

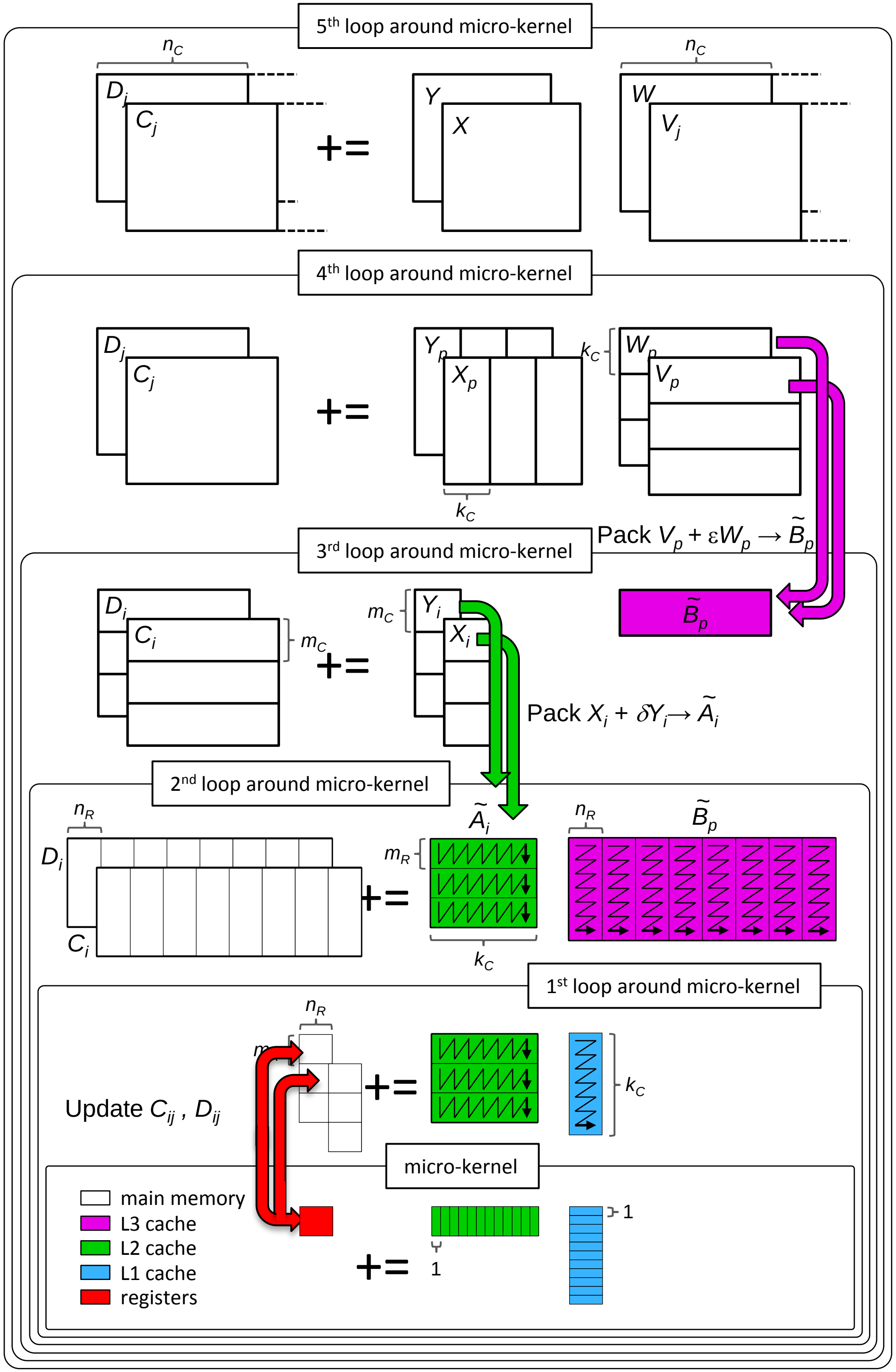
endfor

endfor

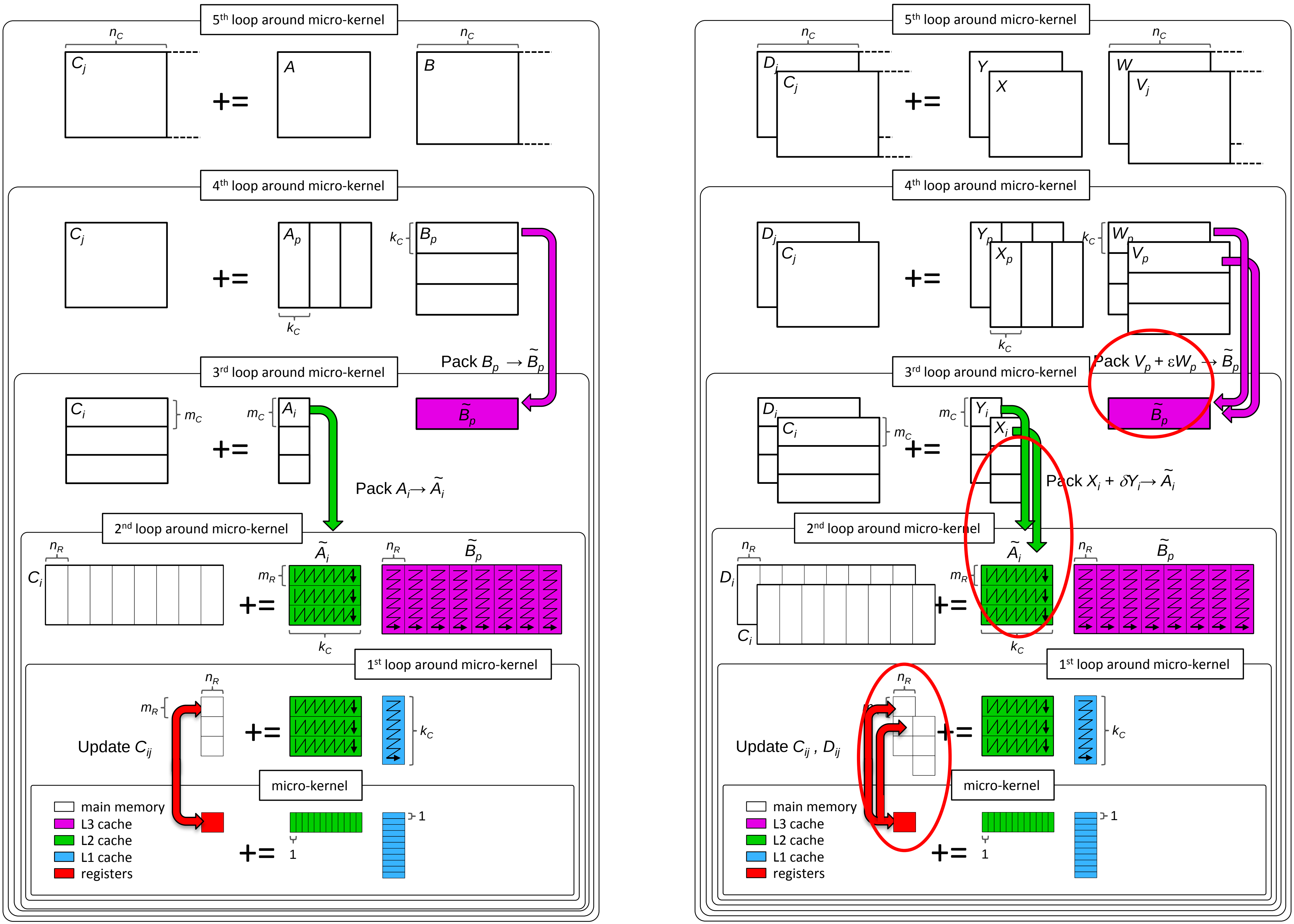
endfor

endfor

endfor



*Jianyu Huang, Tyler Smith, Greg Henry, and Robert van de Geijn. "Strassen's Algorithm Reloaded." In *SC'16*.


5th loop around micro-kernel
4th loop around micro-kernel
3rd loop around micro-kernel
2nd loop around micro-kernel
1st loop around micro-kernel
micro-kernel
Pack $B_p \rightarrow \tilde{B}_p$
Pack $A_i \rightarrow \tilde{A}_i$
Update C_{ij}
Pack $V_p + \varepsilon W_p \rightarrow \tilde{B}_p$
Pack $X_i + \delta Y_i \rightarrow \tilde{A}_i$
Update C_{ij} , D_{ij}
main memory
L3 cache
L2 cache
L1 cache
registers

Two-level Strassen's Algorithm Reloaded

Assume m, n, and k are all multiples of 4. Letting

$$C = \left(\begin{array}{cc|cc} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ \hline C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{array}\right), A = \left(\begin{array}{cc|cc} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ \hline A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{array}\right), B = \left(\begin{array}{cc|cc} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ \hline B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{array}\right),$$

where $C_{i,j}$ is $\frac{m}{4} \times \frac{n}{4}$, $A_{i,p}$ is $\frac{m}{4} \times \frac{k}{4}$, and $B_{p,j}$ is $\frac{k}{4} \times \frac{n}{4}$.

Two-level Strassen's Algorithm Reloaded (Continue)

$M_0 := \alpha(A_{0,0}+A_{2,2}+A_{1,1}+A_{3,3})(B_{0,0}+B_{2,2}+B_{1,1}+B_{3,3});$	$C_{0,0} += M_0;$	$C_{1,1} += M_0;$	$C_{2,2} += M_0;$	$C_{3,3} += M_0;$
$M_1 := \alpha(A_{1,0}+A_{3,2}+A_{1,1}+A_{3,3})(B_{0,0}+B_{2,2});$	$C_{1,0} += M_1;$	$C_{1,1} -= M_1;$	$C_{3,2} += M_1;$	$C_{3,3} -= M_1;$
$M_2 := \alpha(A_{0,0}+A_{2,2})(B_{0,1}+B_{2,3}+B_{1,1}+B_{3,3});$	$C_{0,1} += M_2;$	$C_{1,1} += M_2;$	$C_{2,3} += M_2;$	$C_{3,3} += M_2;$
$M_3 := \alpha(A_{1,1}+A_{3,3})(B_{1,0}+B_{3,2}+B_{0,0}+B_{2,2});$	$C_{0,0} += M_3;$	$C_{1,0} += M_3;$	$C_{2,2} += M_3;$	$C_{3,2} += M_3;$
$M_4 := \alpha(A_{0,0}+A_{2,2}+A_{0,1}+A_{2,3})(B_{1,1}+B_{3,3});$	$C_{0,0} -= M_4;$	$C_{0,1} += M_4;$	$C_{2,2} -= M_4;$	$C_{2,3} += M_4;$
$M_5 := \alpha(A_{1,0}+A_{3,2}+A_{0,0}+A_{2,2})(B_{0,0}+B_{2,2}+B_{0,1}+B_{2,3});$	$C_{1,1} += M_5;$	$C_{3,3} += M_5;$		
$M_6 := \alpha(A_{0,1}+A_{2,3}+A_{1,1}+A_{3,3})(B_{1,0}+B_{3,2}+B_{1,1}+B_{3,3});$	$C_{0,0} += M_6;$	$C_{2,2} += M_6;$		
$M_7 := \alpha(A_{2,0}+A_{2,2}+A_{3,1}+A_{3,3})(B_{0,0}+B_{1,1});$	$C_{2,0} += M_7;$	$C_{3,1} += M_7;$	$C_{2,2} -= M_7;$	$C_{3,3} -= M_7;$
$M_8 := \alpha(A_{3,0}+A_{3,2}+A_{3,1}+A_{3,3})(B_{0,0});$	$C_{3,0} += M_8;$	$C_{3,1} -= M_8;$	$C_{3,2} -= M_8;$	$C_{3,3} += M_8;$
$M_9 := \alpha(A_{2,0}+A_{2,2})(B_{0,1}+B_{1,1});$	$C_{2,1} += M_9;$	$C_{3,1} += M_9;$	$C_{2,3} -= M_9;$	$C_{3,3} -= M_9;$
$M_{10} := \alpha(A_{3,1}+A_{3,3})(B_{1,0}+B_{0,0});$	$C_{2,0} += M_{10};$	$C_{3,0} += M_{10};$	$C_{2,2} -= M_{10};$	$C_{3,2} -= M_{10};$
$\vdots$				
$M_{40} := \alpha(A_{3,0}+A_{1,0}+A_{2,0}+A_{0,0})(B_{0,0}+B_{0,2}+B_{0,1}+B_{0,3});$	$C_{3,3} += M_{40};$			
$M_{41} := \alpha(A_{2,1}+A_{0,1}+A_{3,1}+A_{1,1})(B_{1,0}+B_{1,2}+B_{1,1}+B_{1,3});$	$C_{2,2} += M_{41};$			
$M_{42} := \alpha(A_{0,2}+A_{2,2}+A_{1,3}+A_{3,3})(B_{2,0}+B_{2,2}+B_{3,1}+B_{3,3});$	$C_{0,0} += M_{42};$	$C_{1,1} += M_{42};$		
$M_{43} := \alpha(A_{1,2}+A_{3,2}+A_{1,3}+A_{3,3})(B_{2,0}+B_{2,2});$	$C_{1,0} += M_{43};$	$C_{1,1} -= M_{43};$		
$M_{44} := \alpha(A_{0,2}+A_{2,2})(B_{2,1}+B_{2,3}+B_{3,1}+B_{3,3});$	$C_{0,1} += M_{44};$	$C_{1,1} += M_{44};$		
$M_{45} := \alpha(A_{1,3}+A_{3,3})(B_{3,0}+B_{3,2}+B_{2,0}+B_{2,2});$	$C_{0,0} += M_{45};$	$C_{1,0} += M_{45};$		
$M_{46} := \alpha(A_{0,2}+A_{2,2}+A_{0,3}+A_{2,3})(B_{3,1}+B_{3,3});$	$C_{0,0} -= M_{46};$	$C_{0,1} += M_{46};$		
$M_{47} := \alpha(A_{1,2}+A_{3,2}+A_{0,2}+A_{2,2})(B_{2,0}+B_{2,2}+B_{2,1}+B_{2,3});$	$C_{1,1} += M_{47};$			
$M_{48} := \alpha(A_{0,3}+A_{2,3}+A_{1,3}+A_{3,3})(B_{3,0}+B_{3,2}+B_{3,1}+B_{3,3});$	$C_{0,0} += M_{48};$			

$M := \alpha(X_0+X_1+X_2+X_3)(V+V_1+V_2+V_3);$ $\quad C_0 += M;$ $\quad C_1 += M;$ $\quad C_2 += M;$ $\quad C_3 += M;$

General operation for two-level Strassen:

$M := \alpha(X_0+\delta_1 X_1+\delta_2 X_2+\delta_3 X_3)(V+\varepsilon_1 V_1+\varepsilon_2 V_2+\varepsilon_3 V_3);$ $\quad C_0 += \gamma_0 M;$ $\quad C_1 += \gamma_1 M;$ $\quad C_2 += \gamma_2 M;$ $\quad C_3 += \gamma_3 M;$

$\gamma_i, \delta_i, \varepsilon_i \in \{-1, 0, 1\}.$

Additional Levels of Strassen Reloaded

- The general operation of one-level Strassen:

$M := \alpha(X + \delta Y)(V + \varepsilon W); \qquad C += \gamma_0 M;\ D += \gamma_1 M;$
$\gamma_0, \gamma_1, \delta, \varepsilon \in \{\text{-}1, 0, 1\}.$

- The general operation of two-level Strassen:

$M := \alpha(X_0 + \delta_1 X_1 + \delta_2 X_2 + \delta_3 X_3)(V + \varepsilon_1 V_1 + \varepsilon_2 V_2 + \varepsilon_3 V_3);$
$C_0 += \gamma_0 M;\ C_1 += \gamma_1 M;\ C_2 += \gamma_2 M;\ C_3 += \gamma_3 M;$
$\gamma_i, \delta_i, \varepsilon_i \in \{\text{-}1, 0, 1\}.$

- The general operation needed to integrate k levels of Strassen is given by

$M := \alpha \left(\sum_{s=0}^{l_X - 1} \delta_s X_s \right) \left(\sum_{t=0}^{l_V - 1} \epsilon_t V_t \right);$
$C_r += \gamma_r M \quad \text{for } r = 0, \ldots, l_C - 1;$
$\delta_i, \epsilon_i, \gamma_i \in \{-1, 0, 1\}.$

Building blocks

$$M := \alpha \left(\sum_{s=0}^{l_X-1} \delta_s X_s\right)\left(\sum_{t=0}^{l_V-1} \epsilon_t V_t\right);$$
$$C_r += \gamma_r M \quad \text{for } r = 0, \ldots, l_C - 1;$$
$$\delta_i, \epsilon_i, \gamma_i \in \{-1, 0, 1\}.$$

BLIS framework

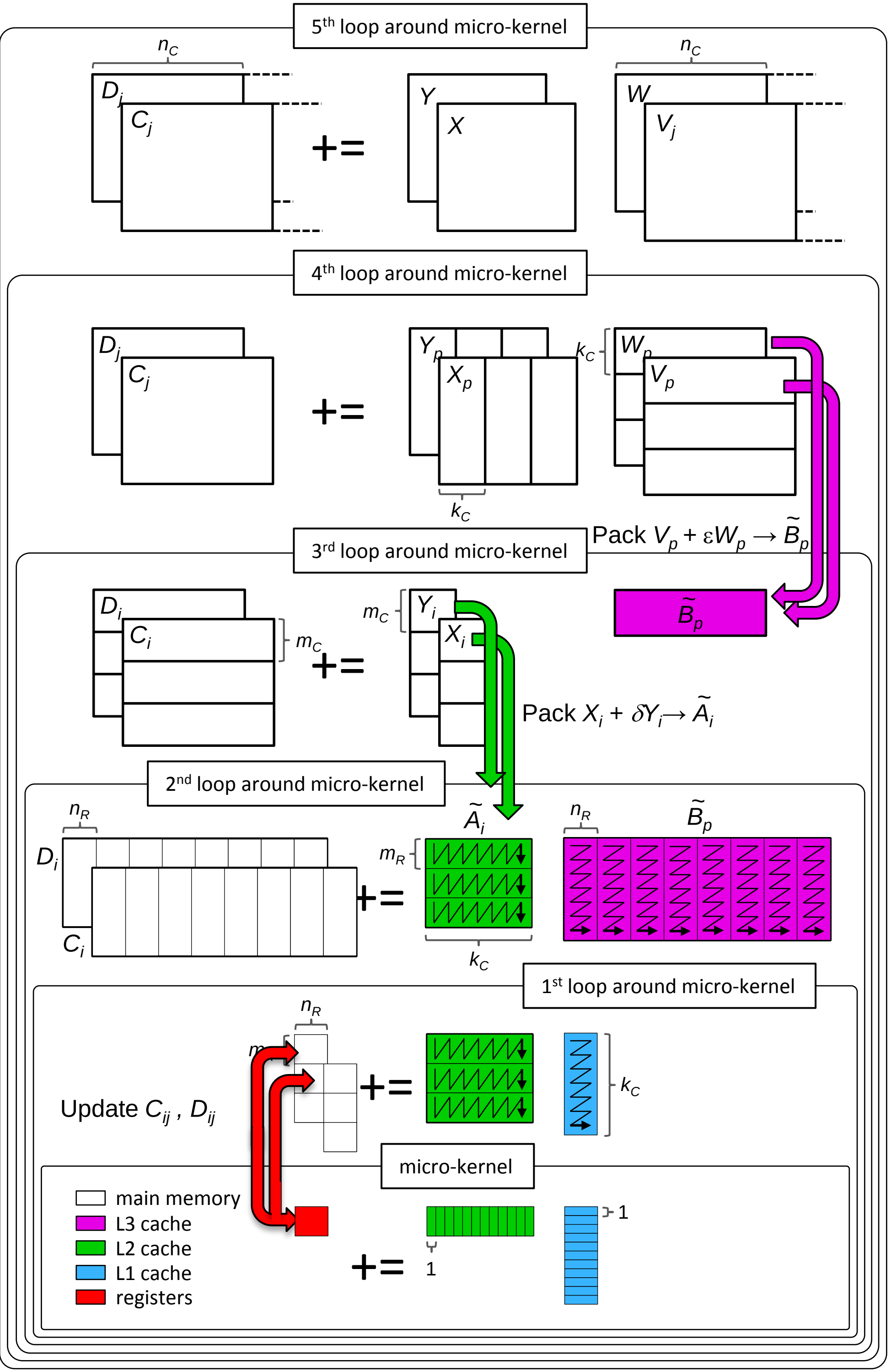
- A routine for packing B_p into $\widetilde{B}_p$
  - written in C/Intel intrinsics
- A routine for packing A_i into $\widetilde{A}_i$
  - written in C/Intel intrinsics
- A micro-kernel for updating an $m_R \times n_R$ submatrix of C.
  - written in SIMD assembly (AVX, FMA, AVX512, etc)

Adapted to general operation

- Integrate the addition of multiple matrices V_t into $\widetilde{B}_p$
- Integrate the addition of multiple matrices X_s into $\widetilde{A}_i$
- Integrate the update of multiple submatrices of C.

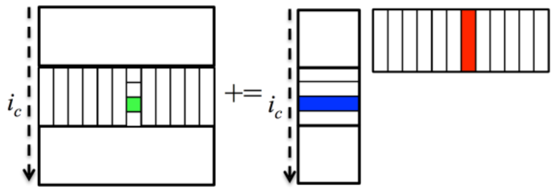
Variations on a theme

- Naïve Strassen
  - A traditional implementation with temporary buffers.
- AB Strassen
  - Integrate the addition of matrices into $\widetilde{A}_i$ and $\widetilde{B}_p$.
- ABC Strassen
  - Integrate the addition of matrices into $\widetilde{A}_i$ and $\widetilde{B}_p$.
  - Integrate the update of multiple submatrices of C in the micro-kernel.

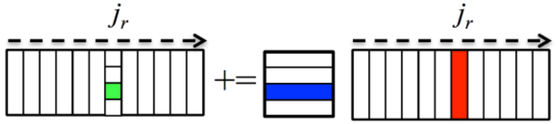


Parallelization

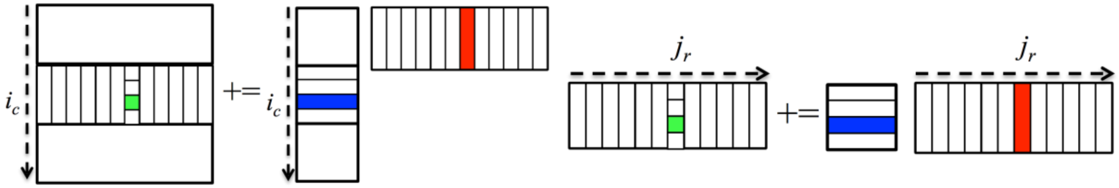
- 3rd loop (along m_C direction)



- 2nd loop (along n_R direction)



- both 3rd and 2nd loop



*Tyler M. Smith, Robert Van De Geijn, Mikhail Smelyanskiy, Jeff R. Hammond, and Field G. Van Zee. "Anatomy of high-performance many-threaded matrix multiplication." In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pp. 1049-1059. IEEE, 2014.

Outline

- Standard Matrix-matrix multiplication
- Strassen’s Algorithm Reloaded
- Theoretical Model and Analysis
- Performance Experiments
- Conclusion

Performance Model

- Performance Metric

$$\text{Effective GFLOPS} = \frac{2 \cdot m \cdot n \cdot k}{\text{time (in seconds)}} \cdot 10^{-9}$$

- Total Time Breakdown

$$T = T_a + T_m$$

Arithmetic Operations

Memory Operations

Arithmetic Operations

$$T_a = T_a^{\times} + T_a^{A+} + T_a^{B+} + T_a^{C+}$$

$T_a^{\times}$: Submatrix multiplication

T_a^{B+}: Extra additions with submatrices of A, B, C, respectively

- DGEMM
  - No extra additions

$$T_a = 2mnk \cdot \tau_a$$

- One-level Strassen (ABC, AB, Naïve)
  - 7 submatrix multiplications
  - 5 extra additions of submatrices of A and B
  - 12 extra additions of submatrices of C

$$T_a = (7 \times 2\frac{m}{2}\frac{n}{2}\frac{k}{2} + 5 \times 2\frac{m}{2}\frac{k}{2} + 5 \times 2\frac{k}{2}\frac{n}{2} + 12 \times 2\frac{m}{2}\frac{n}{2}) \cdot \tau_a$$

$$\begin{array}{ll}
M_0 := \alpha(A_{00}+A_{11})(B_{00}+B_{11}); & C_{00} \mathrel{+}= M_0;\ C_{11} \mathrel{+}= M_0; \\
M_1 := \alpha(A_{10}+A_{11})B_{00}; & C_{10} \mathrel{+}= M_1;\ C_{11} \mathrel{-}= M_1; \\
M_2 := \alpha A_{00}(B_{01}-B_{11}); & C_{01} \mathrel{+}= M_2;\ C_{11} \mathrel{+}= M_2; \\
M_3 := \alpha A_{11}(B_{10}-B_{00}); & C_{00} \mathrel{+}= M_3;\ C_{10} \mathrel{+}= M_3; \\
M_4 := \alpha(A_{00}+A_{01})B_{11}; & C_{01} \mathrel{+}= M_4;\ C_{00} \mathrel{-}= M_4; \\
M_5 := \alpha(A_{10}-A_{00})(B_{00}+B_{01}); & C_{11} \mathrel{+}= M_5; \\
M_6 := \alpha(A_{01}-A_{11})(B_{10}+B_{11}); & C_{00} \mathrel{+}= M_6;
\end{array}$$

- Two-level Strassen (ABC, AB, Naïve)
  - 49 submatrix multiplications
  - 95 extra additions of submatrices of A and B
  - 154 extra additions of submatrices of C

$$T_a = (49 \times 2\frac{m}{4}\frac{n}{4}\frac{k}{4} + 95 \times 2\frac{m}{4}\frac{k}{4} + 95 \times 2\frac{k}{4}\frac{n}{4} + 154 \times 2\frac{m}{4}\frac{n}{4}) \cdot \tau_a$$

$$\begin{array}{lllll}
M_0 = (A_{0,0}+A_{2,2}+A_{1,1}+A_{3,3})(B_{0,0}+B_{2,2}+B_{1,1}+B_{3,3}); & C_{0,0} \mathrel{+}= M_0; & C_{1,1} \mathrel{+}= M_0; & C_{2,2} \mathrel{+}= M_0; & C_{3,3} \mathrel{+}= M_0; \\
M_1 = (A_{1,0}+A_{3,2}+A_{1,1}+A_{3,3})(B_{0,0}+B_{2,2}); & C_{1,0} \mathrel{+}= M_1; & C_{1,1} \mathrel{-}= M_1; & C_{3,2} \mathrel{+}= M_1; & C_{3,3} \mathrel{-}= M_1; \\
M_2 = (A_{0,0}+A_{2,2})(B_{0,1}+B_{2,3}+B_{1,1}+B_{3,3}); & C_{0,1} \mathrel{+}= M_2; & C_{1,1} \mathrel{+}= M_2; & C_{2,3} \mathrel{+}= M_2; & C_{3,3} \mathrel{+}= M_2; \\
M_3 = (A_{1,1}+A_{3,3})(B_{1,0}+B_{3,2}+B_{0,0}+B_{2,2}); & C_{0,0} \mathrel{+}= M_3; & C_{1,0} \mathrel{+}= M_3; & C_{2,2} \mathrel{+}= M_3; & C_{3,2} \mathrel{+}= M_3; \\
M_4 = (A_{0,0}+A_{2,2}+A_{0,1}+A_{2,3})(B_{1,1}+B_{3,3}); & C_{0,0} \mathrel{-}= M_4; & C_{0,1} \mathrel{+}= M_4; & C_{2,2} \mathrel{-}= M_4; & C_{2,3} \mathrel{+}= M_4; \\
M_5 = (A_{1,0}+A_{3,2}+A_{0,0}+A_{2,2})(B_{0,0}+B_{2,2}+B_{0,1}+B_{2,3}); & C_{1,1} \mathrel{+}= M_5; & C_{3,3} \mathrel{+}= M_5; & & \\
M_6 = (A_{0,1}+A_{2,3}+A_{1,1}+A_{3,3})(B_{1,0}+B_{3,2}+B_{1,1}+B_{3,3}); & C_{0,0} \mathrel{+}= M_6; & C_{2,2} \mathrel{+}= M_6; & & \\
M_7 = (A_{2,0}+A_{2,2}+A_{3,1}+A_{3,3})(B_{0,0}+B_{1,1}); & C_{2,0} \mathrel{+}= M_7; & C_{3,1} \mathrel{+}= M_7; & C_{2,2} \mathrel{-}= M_7; & C_{3,3} \mathrel{-}= M_7; \\
M_8 = (A_{3,0}+A_{3,2}+A_{3,1}+A_{3,3})(B_{0,0}); & C_{3,0} \mathrel{+}= M_8; & C_{3,1} \mathrel{-}= M_8; & C_{3,2} \mathrel{-}= M_8; & C_{3,3} \mathrel{+}= M_8; \\
M_9 = (A_{2,0}+A_{2,2})(B_{0,1}+B_{1,1}); & C_{2,1} \mathrel{+}= M_9; & C_{3,1} \mathrel{+}= M_9; & C_{2,3} \mathrel{-}= M_9; & C_{3,3} \mathrel{-}= M_9; \\
M_{10} = (A_{3,1}+A_{3,3})(B_{1,0}+B_{0,0}); & C_{2,0} \mathrel{+}= M_{10}; & C_{3,0} \mathrel{+}= M_{10}; & C_{2,2} \mathrel{-}= M_{10}; & C_{3,2} \mathrel{-}= M_{10}; \\
\vdots & & & & \\
M_{40} = (A_{3,0}+A_{1,0}+A_{2,0}+A_{0,0})(B_{0,0}+B_{0,2}+B_{0,1}+B_{0,3}); & C_{3,3} \mathrel{+}= M_{40}; & & & \\
M_{41} = (A_{2,1}+A_{0,1}+A_{3,1}+A_{1,1})(B_{1,0}+B_{1,2}+B_{1,1}+B_{1,3}); & C_{2,2} \mathrel{+}= M_{41}; & & & \\
M_{42} = (A_{0,2}+A_{2,2}+A_{1,3}+A_{3,3})(B_{2,0}+B_{2,2}+B_{3,1}+B_{3,3}); & C_{0,0} \mathrel{+}= M_{42}; & C_{1,1} \mathrel{+}= M_{42}; & & \\
M_{43} = (A_{1,2}+A_{3,2}+A_{1,3}+A_{3,3})(B_{2,0}+B_{2,2}); & C_{1,0} \mathrel{+}= M_{43}; & C_{1,1} \mathrel{-}= M_{43}; & & \\
M_{44} = (A_{0,2}+A_{2,2})(B_{2,1}+B_{2,3}+B_{3,1}+B_{3,3}); & C_{0,1} \mathrel{+}= M_{44}; & C_{1,1} \mathrel{+}= M_{44}; & & \\
M_{45} = (A_{1,3}+A_{3,3})(B_{3,0}+B_{3,2}+B_{2,0}+B_{2,2}); & C_{0,0} \mathrel{+}= M_{45}; & C_{1,0} \mathrel{+}= M_{45}; & & \\
M_{46} = (A_{0,2}+A_{2,2}+A_{0,3}+A_{2,3})(B_{3,1}+B_{3,3}); & C_{0,0} \mathrel{-}= M_{46}; & C_{0,1} \mathrel{+}= M_{46}; & & \\
M_{47} = (A_{1,2}+A_{3,2}+A_{0,2}+A_{2,2})(B_{2,0}+B_{2,2}+B_{2,1}+B_{2,3}); & C_{1,1} \mathrel{+}= M_{47}; & & & \\
M_{48} = (A_{0,3}+A_{2,3}+A_{1,3}+A_{3,3})(B_{3,0}+B_{3,2}+B_{3,1}+B_{3,3}); & C_{0,0} \mathrel{+}= M_{48}; & & &
\end{array}$$

Memory Operations

$$T_m = N_m^{A\times} \cdot T_m^{A\times} + N_m^{B\times} \cdot T_m^{B\times} + N_m^{C\times} \cdot T_m^{C\times} + N_m^{A+} \cdot T_m^{A+} + N_m^{B+} \cdot T_m^{B+} + N_m^{C+} \cdot T_m^{C+}$$

- DGEMM $T_m = (1 \cdot mk\lceil \frac{n}{n_c} \rceil + 1 \cdot nk + 1 \cdot 2\lambda mn\lceil \frac{k}{k_c} \rceil) \cdot \tau_b$

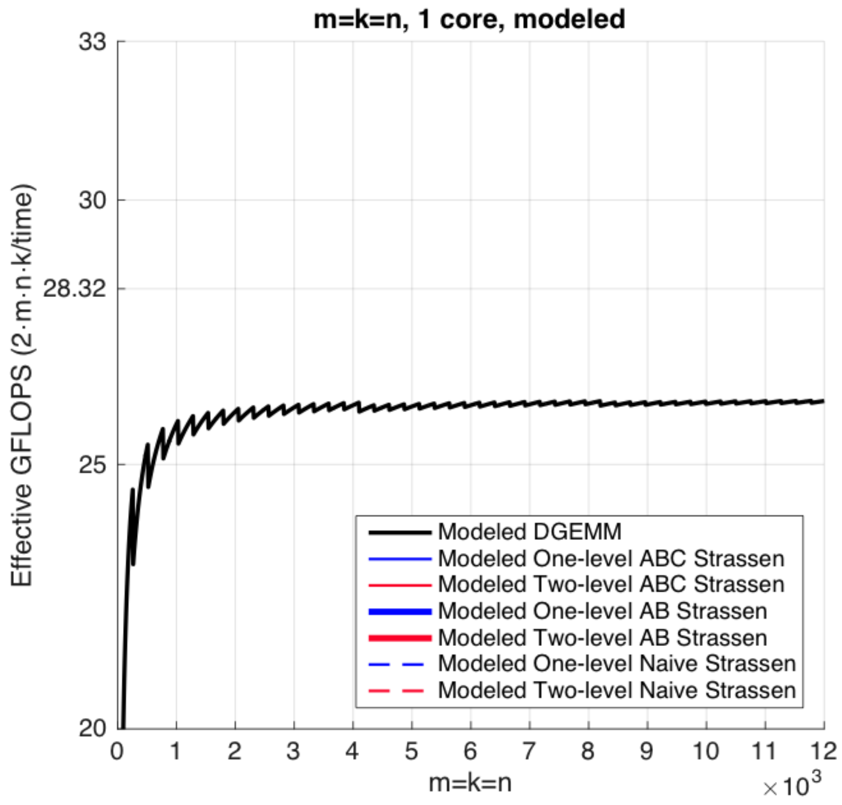
- One-level
  - ABC Strassen $T_m = (12 \cdot \frac{m}{2}\frac{k}{2}\lceil \frac{n/2}{n_c} \rceil + 12 \cdot \frac{n}{2}\frac{k}{2} + 12 \cdot 2\lambda \frac{m}{2}\frac{n}{2}\lceil \frac{k/2}{k_c} \rceil) \cdot \tau_b$
  - AB Strassen $T_m = (12 \cdot \frac{m}{2}\frac{k}{2}\lceil \frac{n/2}{n_c} \rceil + 12 \cdot \frac{n}{2}\frac{k}{2} + 7 \cdot 2\lambda \frac{m}{2}\frac{n}{2}\lceil \frac{k/2}{k_c} \rceil + 36 \cdot \frac{m}{2}\frac{n}{2}) \cdot \tau_b$
  - Naïve Strassen $T_m = (7 \cdot \frac{m}{2}\frac{k}{2}\lceil \frac{n/2}{n_c} \rceil + 7 \cdot \frac{n}{2}\frac{k}{2} + 7 \cdot 2\lambda \frac{m}{2}\frac{n}{2}\lceil \frac{k/2}{k_c} \rceil + 19 \cdot \frac{m}{2}\frac{k}{2} + 19 \cdot \frac{n}{2}\frac{k}{2} + 36 \cdot \frac{m}{2}\frac{n}{2}) \cdot \tau_b$

- Two-level
  - ABC Strassen $T_m = (194 \cdot \frac{m}{4}\frac{k}{4}\lceil \frac{n/4}{n_c} \rceil + 194 \cdot \frac{n}{4}\frac{k}{4} + 154 \cdot 2\lambda \frac{m}{4}\frac{n}{4}\lceil \frac{k/4}{k_c} \rceil) \cdot \tau_b$
  - AB Strassen $T_m = (194 \cdot \frac{m}{4}\frac{k}{4}\lceil \frac{n/4}{n_c} \rceil + 194 \cdot \frac{n}{4}\frac{k}{4} + 49 \cdot 2\lambda \frac{m}{4}\frac{n}{4}\lceil \frac{k/4}{k_c} \rceil + 462 \cdot \frac{m}{4}\frac{n}{4}) \cdot \tau_b$
  - Naïve Strassen $T_m = (49 \cdot \frac{m}{4}\frac{k}{4}\lceil \frac{n/4}{n_c} \rceil + 49 \cdot \frac{n}{4}\frac{k}{4} + 49 \cdot 2\lambda \frac{m}{4}\frac{n}{4}\lceil \frac{k/4}{k_c} \rceil + 293 \cdot \frac{m}{4}\frac{k}{4} + 293 \cdot \frac{n}{4}\frac{k}{4} + 462 \cdot \frac{m}{4}\frac{n}{4}) \cdot \tau_b$

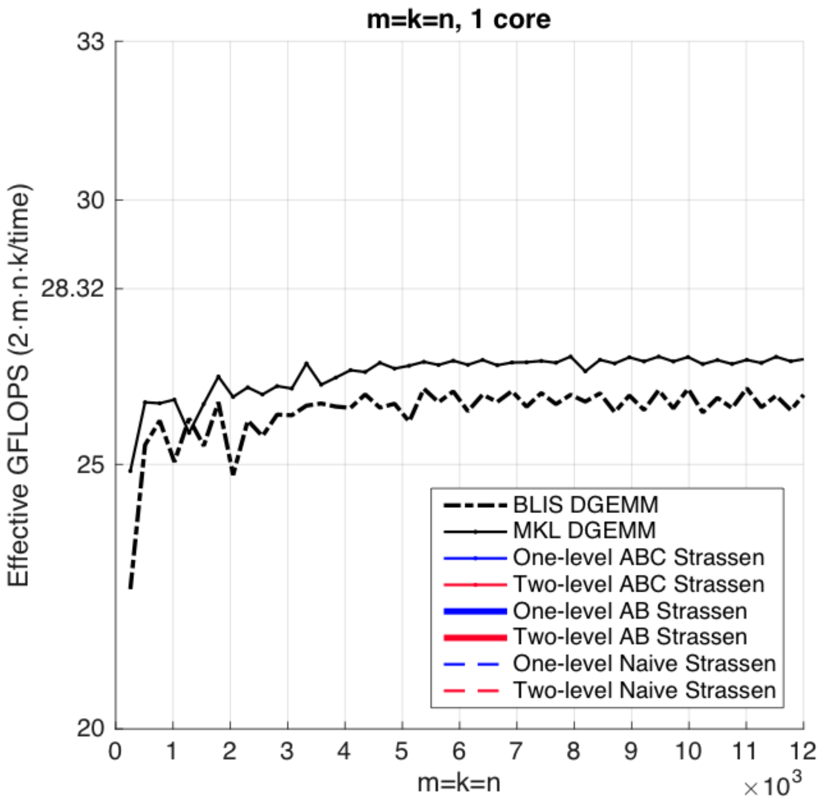
Modeled and Actual Performance on Single Core

Observation (Square Matrices)

Modeled Performance

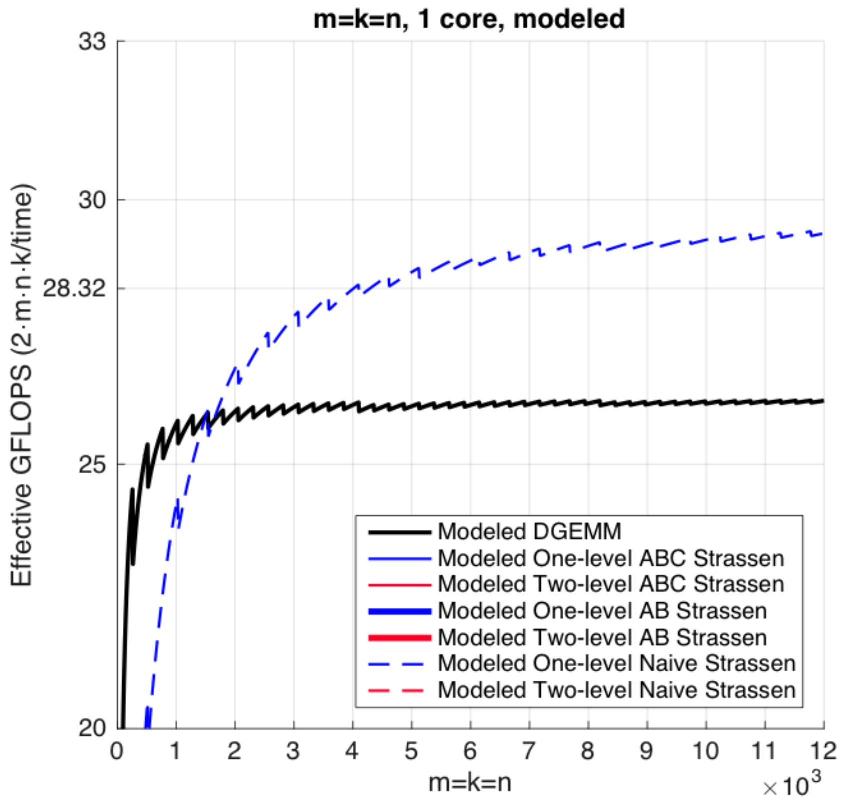


Actual Performance

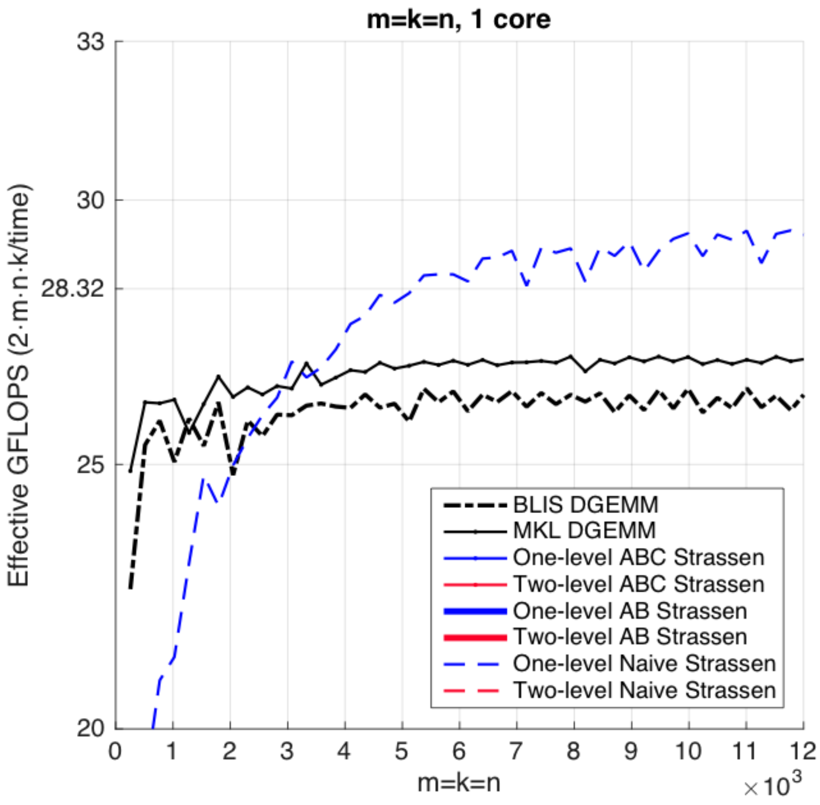


Observation (Square Matrices)

Modeled Performance

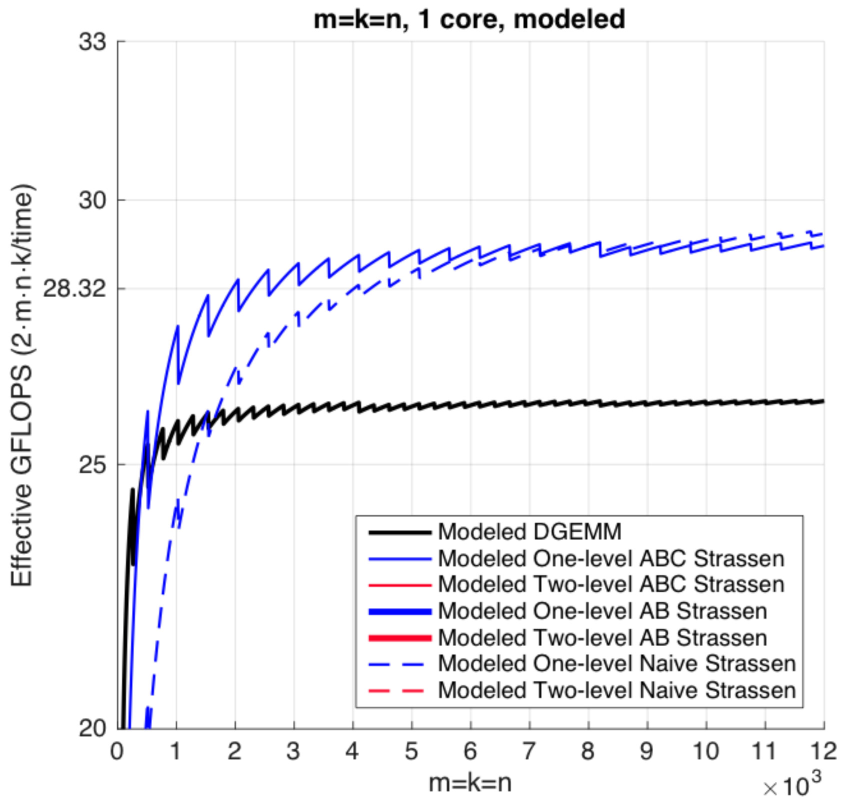


Actual Performance

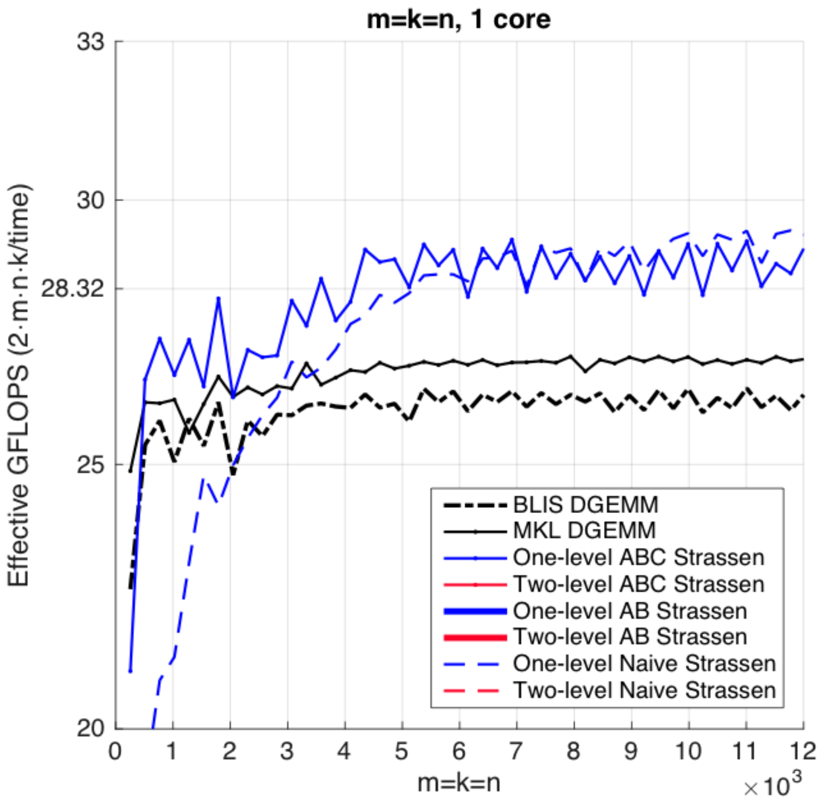


Observation (Square Matrices)

Modeled Performance

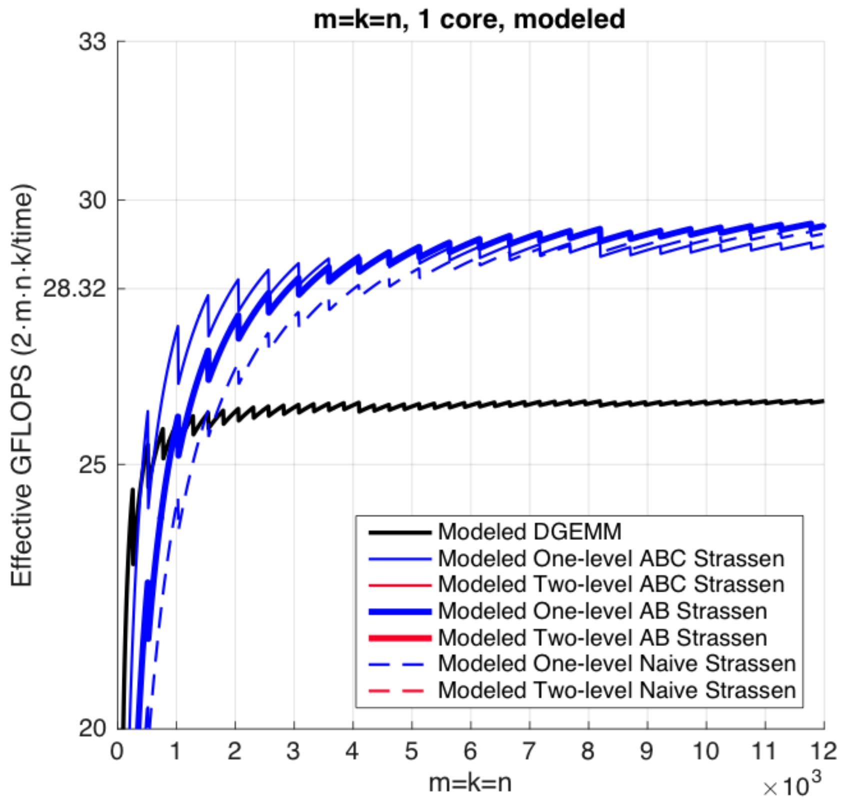


Actual Performance

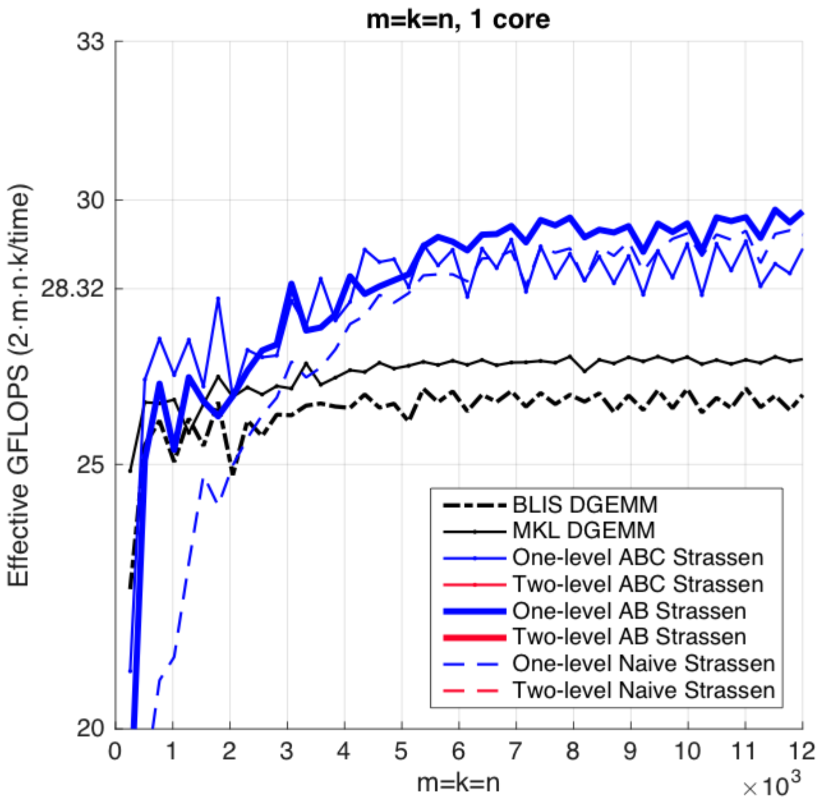


Observation (Square Matrices)

Modeled Performance

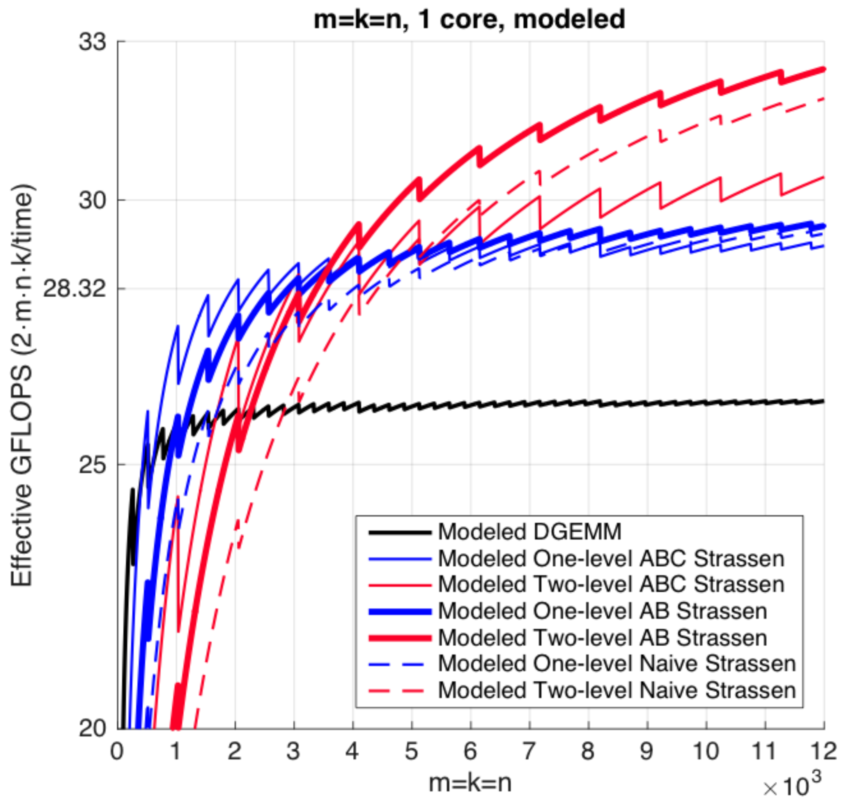


Actual Performance

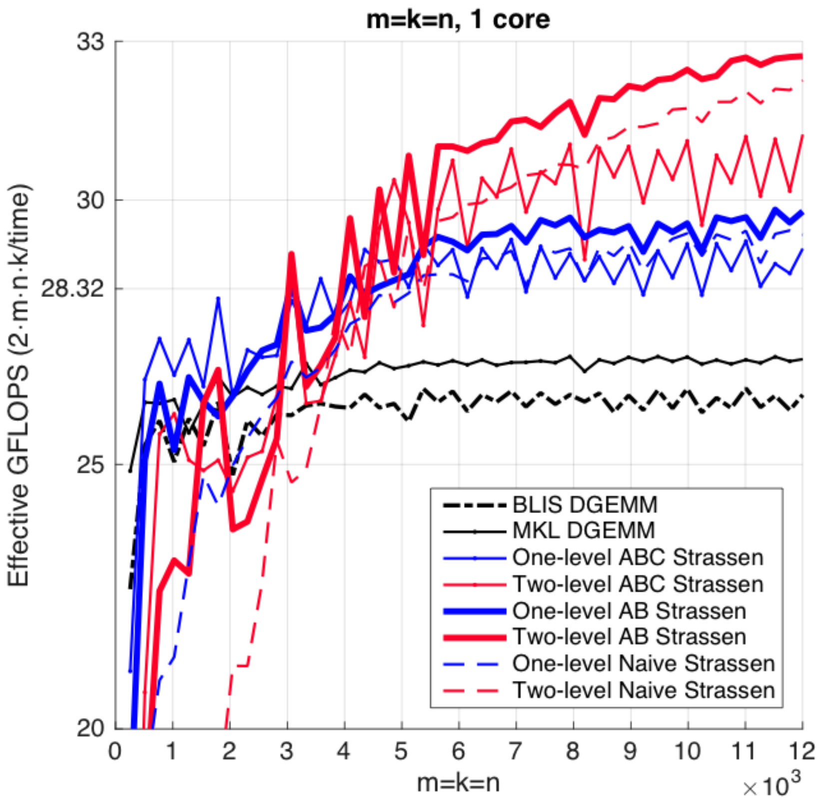


Observation (Square Matrices)

Modeled Performance

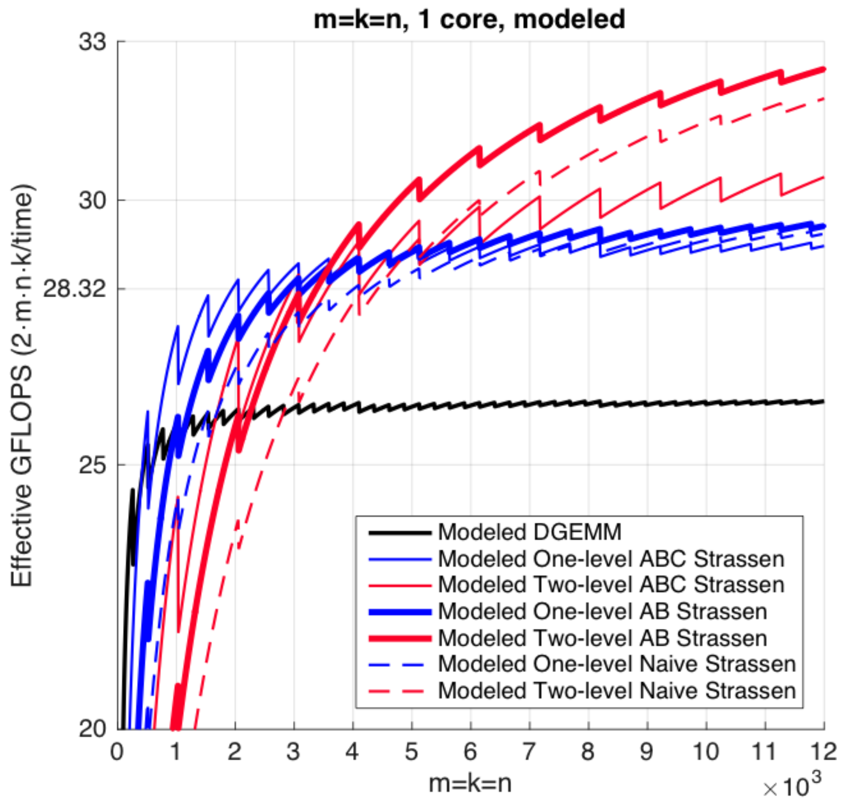


Actual Performance

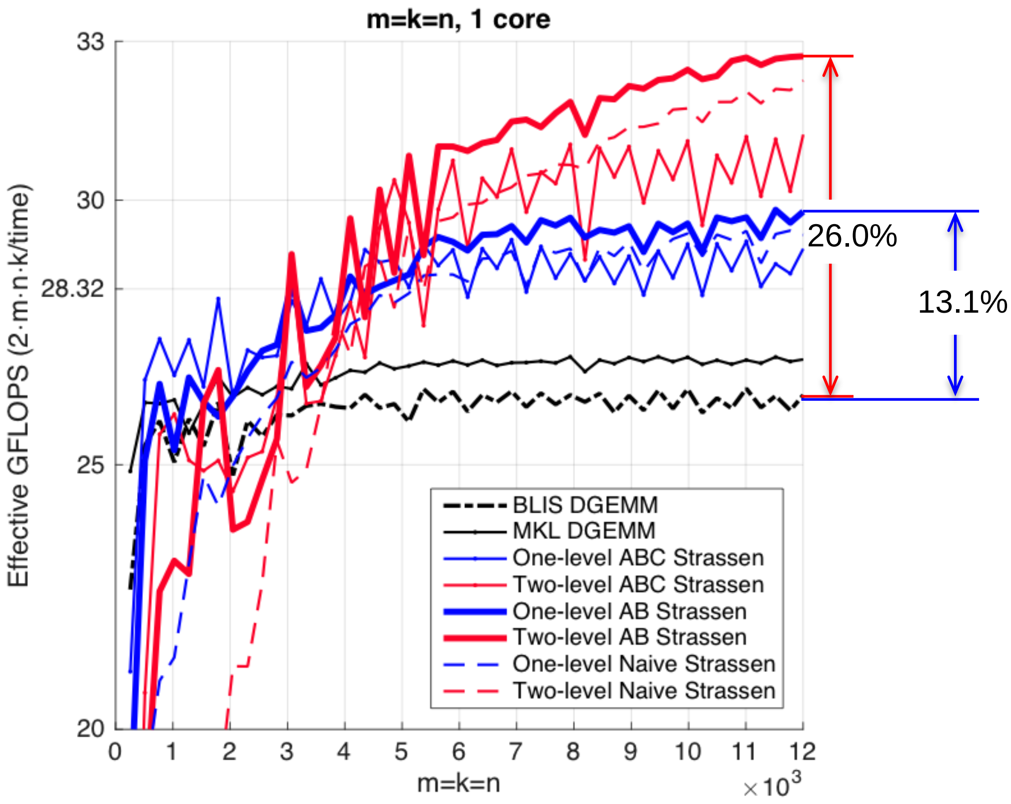


Observation (Square Matrices)

Modeled Performance



Actual Performance

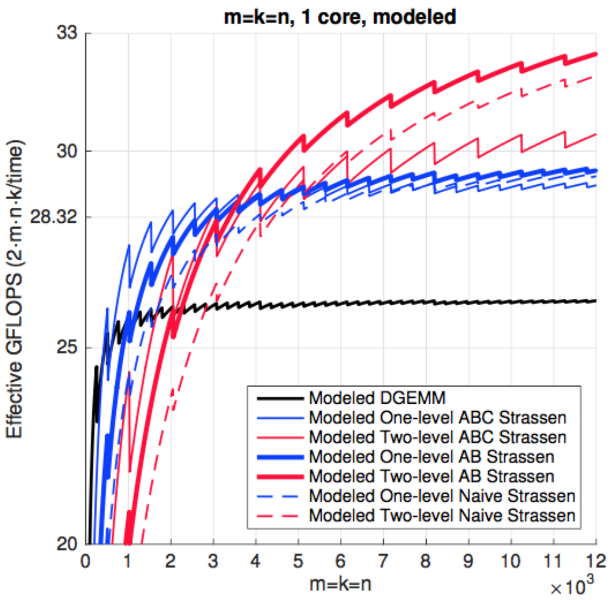


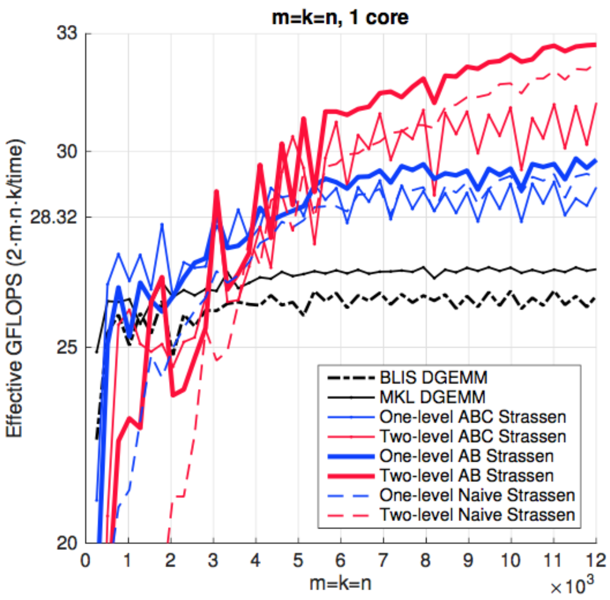
Theoretical Speedup over DGEMM

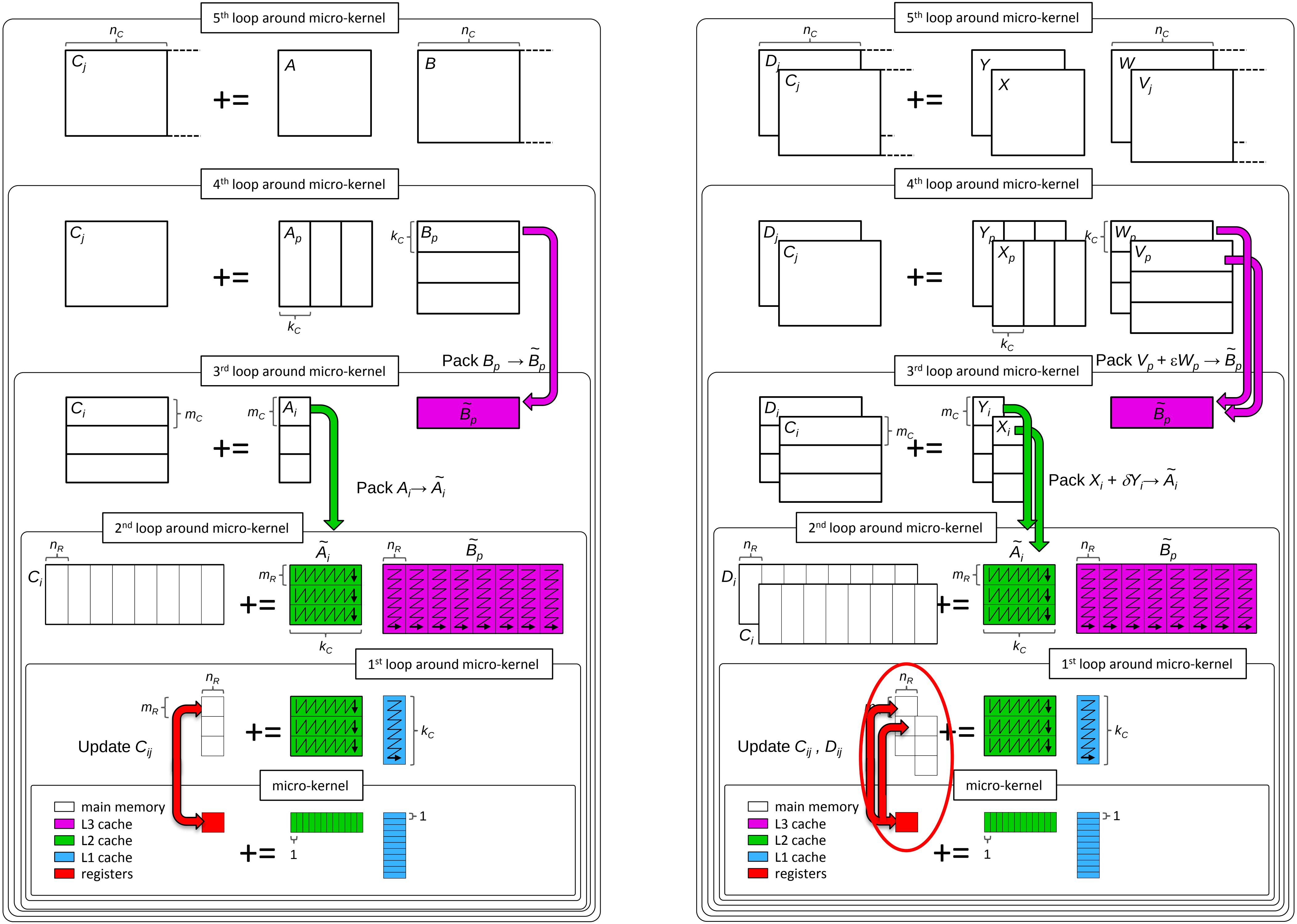
- One-level Strassen (1+14.3% speedup)
  - 8 multiplications → 7 multiplications;
- Two-level Strassen (1+30.6% speedup)
  - 64 multiplications → 49 multiplications;

Observation (Square Matrices)

- Both one-level and two-level
  - ➢ For small square matrices, **ABC Strassen** outperforms **AB Strassen**
  - ➢ For larger square matrices, this trend reverses
- Reason
  - ➢ **ABC Strassen** avoids storing M (M resides in the register)
  - ➢ **ABC Strassen** increases the number of times for updating submatrices of C



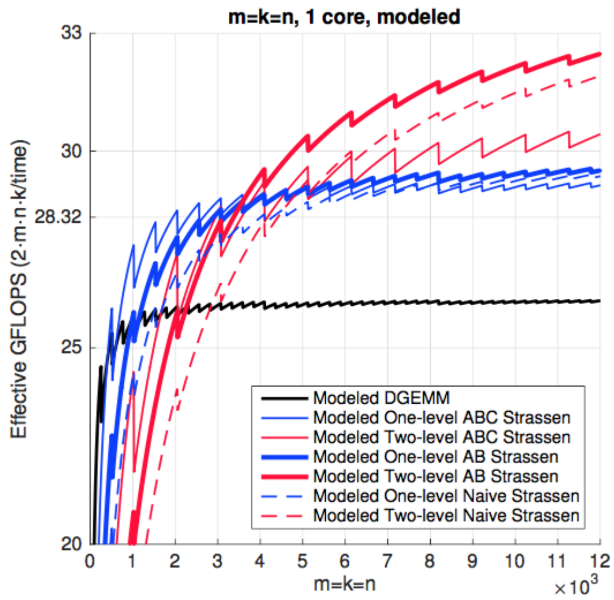


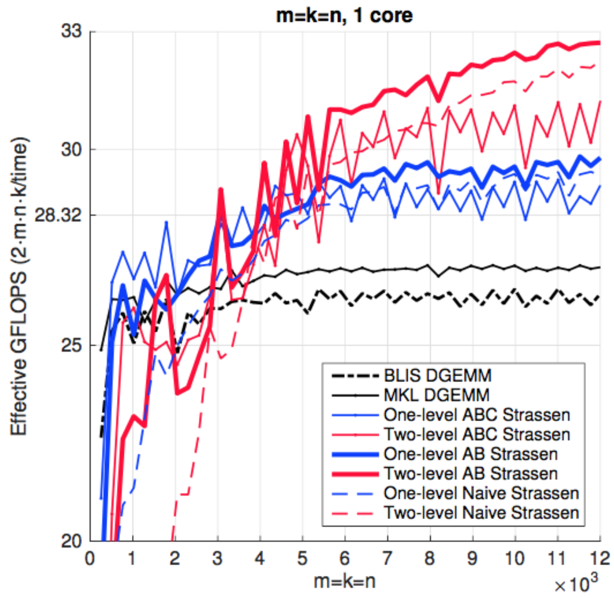


5th loop around micro-kernel
n_C
C_j
+=
A
B
4th loop around micro-kernel
C_j
A_p
B_p
k_C
Pack B_p → B̃_p
3rd loop around micro-kernel
C_i
m_C
A_i
B̃_p
Pack A_i → Ã_i
2nd loop around micro-kernel
n_R
Ã_i
B̃_p
m_R
k_C
1st loop around micro-kernel
Update C_ij
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
1
5th loop around micro-kernel
D_j
C_j
Y
X
W
V_j
4th loop around micro-kernel
Y_p
X_p
W_p
V_p
Pack V_p + εW_p → B̃_p
3rd loop around micro-kernel
D_i
Y_i
X_i
Pack X_i + δY_i → Ã_i
2nd loop around micro-kernel
Update C_ij , D_ij

Observation (Square Matrices)

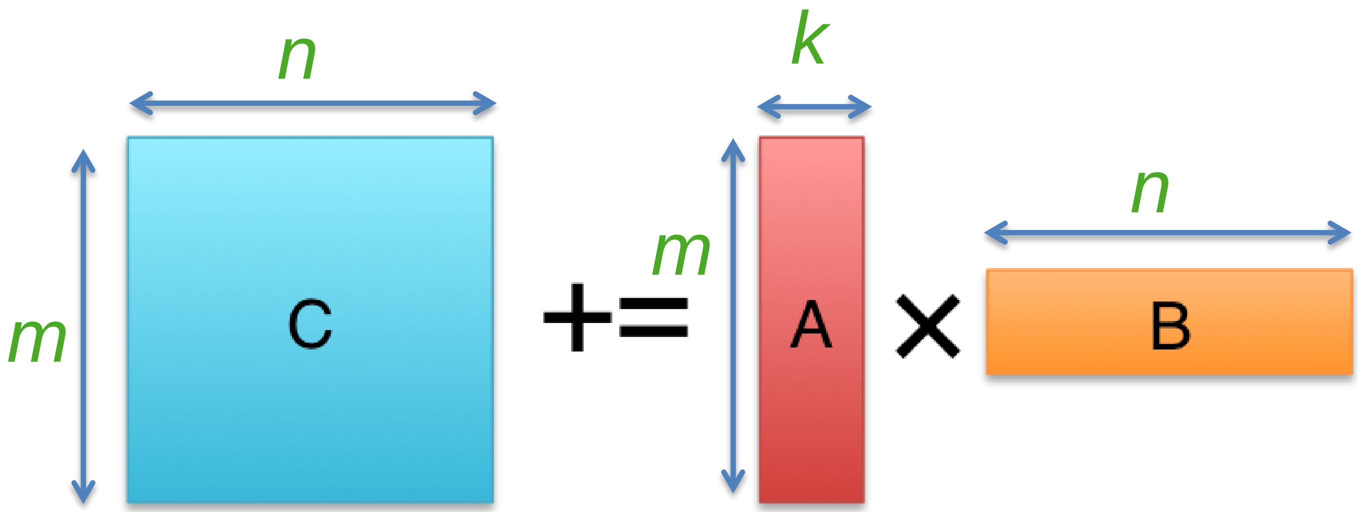
- Both one-level and two-level
  - ➢ For small square matrices, **ABC Strassen** outperforms **AB Strassen**
  - ➢ For larger square matrices, this trend reverses
- Reason
  - ➢ **ABC Strassen** avoids storing M (M resides in the register) 
  - ➢ **ABC Strassen** increases the number of times for updating submatrices of C 





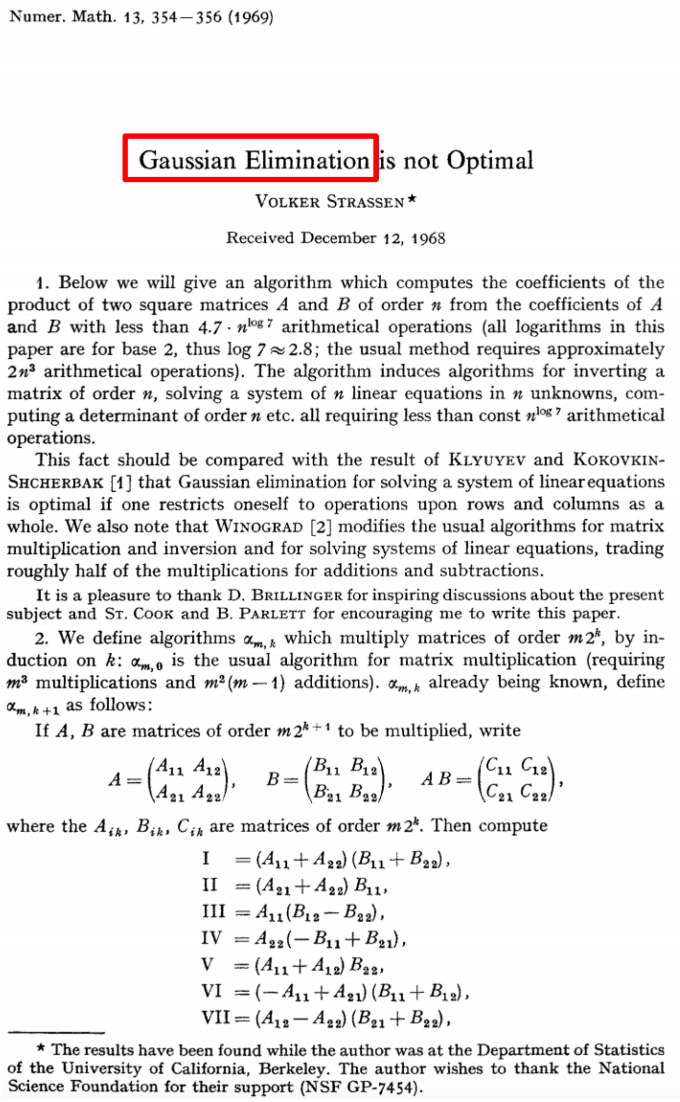
Observation (Rank-k Update)

- What is Rank-k update?



Observation (Rank-k Update)

- Importance of Rank-k update



Numer. Math. 13, 354—356 (1969)

Gaussian Elimination is not Optimal

VOLKER STRASSEN*

Received December 12, 1968

1. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with less than $4.7 \cdot n^{\log 7}$ arithmetical operations (all logarithms in this paper are for base 2, thus $\log 7 \approx 2.8$; the usual method requires approximately $2n^3$ arithmetical operations). The algorithm induces algorithms for inverting a matrix of order n, solving a system of n linear equations in n unknowns, computing a determinant of order n etc. all requiring less than const $n^{\log 7}$ arithmetical operations.

This fact should be compared with the result of KLYUYEV and KOKOVKIN-SHCHERBAK [1] that Gaussian elimination for solving a system of linear equations is optimal if one restricts oneself to operations upon rows and columns as a whole. We also note that WINOGRAD [2] modifies the usual algorithms for matrix multiplication and inversion and for solving systems of linear equations, trading roughly half of the multiplications for additions and subtractions.

It is a pleasure to thank D. BRILLINGER for inspiring discussions about the present subject and ST. COOK and B. PARLETT for encouraging me to write this paper.

2. We define algorithms $\alpha_{m,k}$ which multiply matrices of order $m2^k$, by induction on k: $\alpha_{m,0}$ is the usual algorithm for matrix multiplication (requiring m^3 multiplications and $m^2(m-1)$ additions). $\alpha_{m,k}$ already being known, define $\alpha_{m,k+1}$ as follows:

If A, B are matrices of order $m2^{k+1}$ to be multiplied, write

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad AB = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

where the A_{ik}, B_{ik}, C_{ik} are matrices of order $m2^k$. Then compute

$$\begin{aligned} \mathrm{I} &= (A_{11}+A_{22})(B_{11}+B_{22}), \\ \mathrm{II} &= (A_{21}+A_{22})B_{11}, \\ \mathrm{III} &= A_{11}(B_{12}-B_{22}), \\ \mathrm{IV} &= A_{22}(-B_{11}+B_{21}), \\ \mathrm{V} &= (A_{11}+A_{12})B_{22}, \\ \mathrm{VI} &= (-A_{11}+A_{21})(B_{11}+B_{12}), \\ \mathrm{VII} &= (A_{12}-A_{22})(B_{21}+B_{22}), \end{aligned}$$

* The results have been found while the author was at the Department of Statistics of the University of California, Berkeley. The author wishes to thank the National Science Foundation for their support (NSF GP-7454).

Blocked LU with partial pivoting (**getrf**)

Algorithm: $[A, p] := \text{LUPIV\_BLK}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$, $p \rightarrow \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right)$

where A_{TL} is 0×0, p_T has 0 elements

while $n(A_{TL}) < n(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right), \quad \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \rightarrow \left(\begin{array}{c} p_0 \\ \hline p_1 \\ \hline p_2 \end{array}\right)$$

where A_{11} **is** $b \times b$ **,** p_1 **is** $b \times 1$

$$\left[\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right), p_1\right] := \text{LUPIV\_UNB}\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right)$$

$$\left(\begin{array}{c|c} A_{10} & A_{12} \\ A_{20} & A_{22} \end{array}\right) := \text{PIV}\left(p_1, \left(\begin{array}{c|c} A_{10} & A_{12} \\ A_{20} & A_{22} \end{array}\right)\right)$$

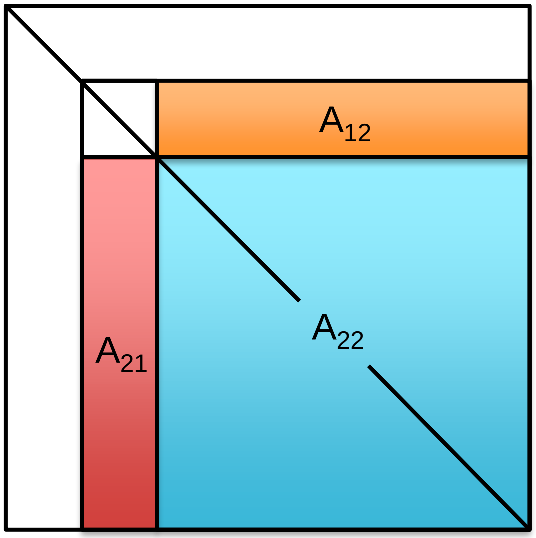
$A_{12} := L_{11}^{-1} A_{12}$

$\boxed{A_{22} := A_{22} - A_{21} A_{12}}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right), \quad \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \leftarrow \left(\begin{array}{c} p_0 \\ \hline p_1 \\ \hline p_2 \end{array}\right)$$

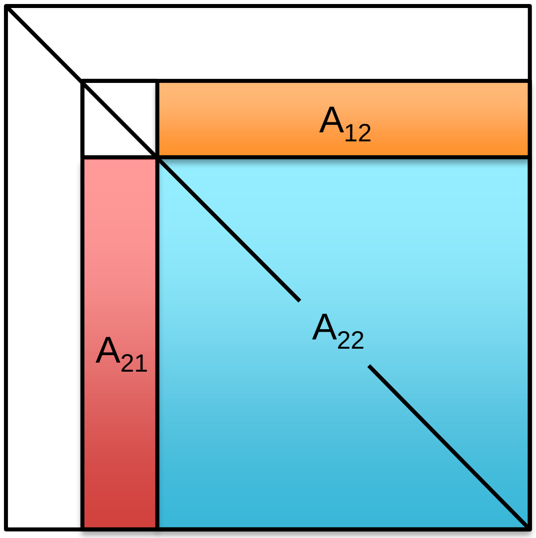
endwhile



Observation (Rank-k Update)

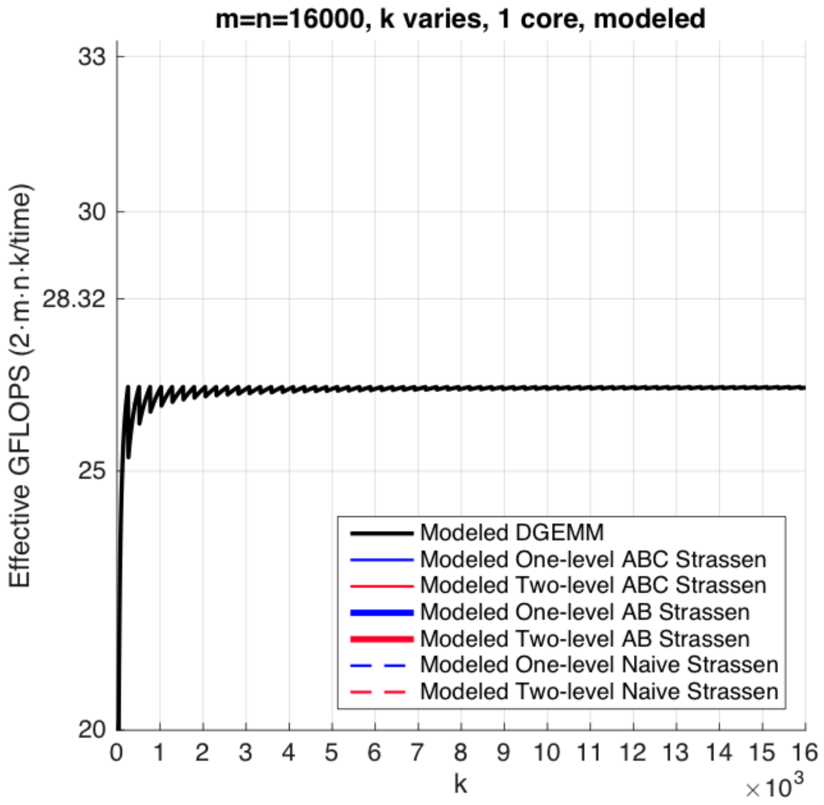
- Importance of Rank-k update

$+= \quad \times$

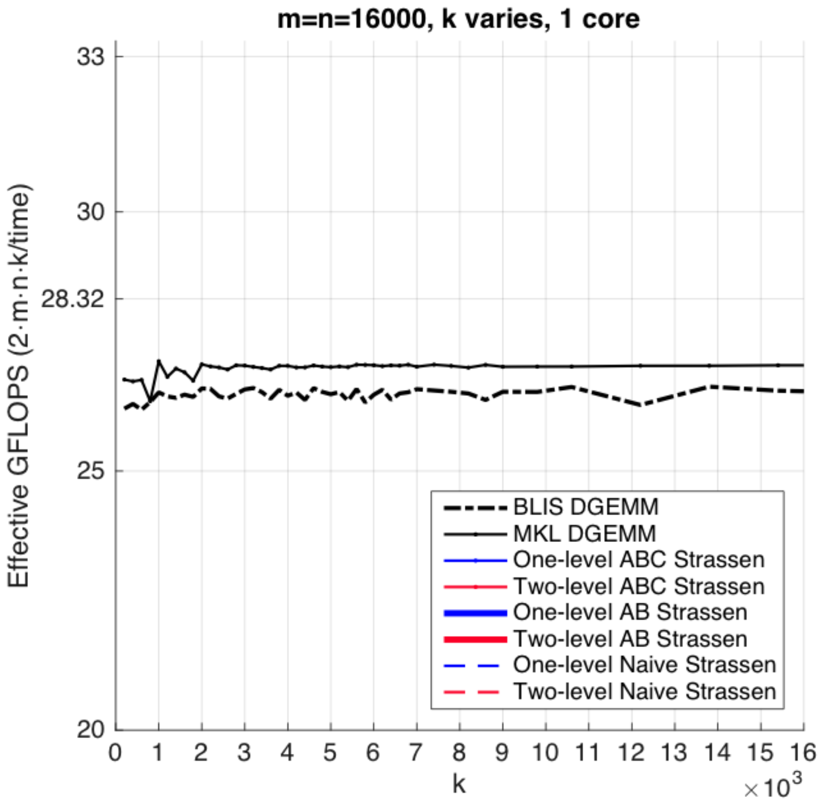


Observation (Rank-k Update)

Modeled Performance

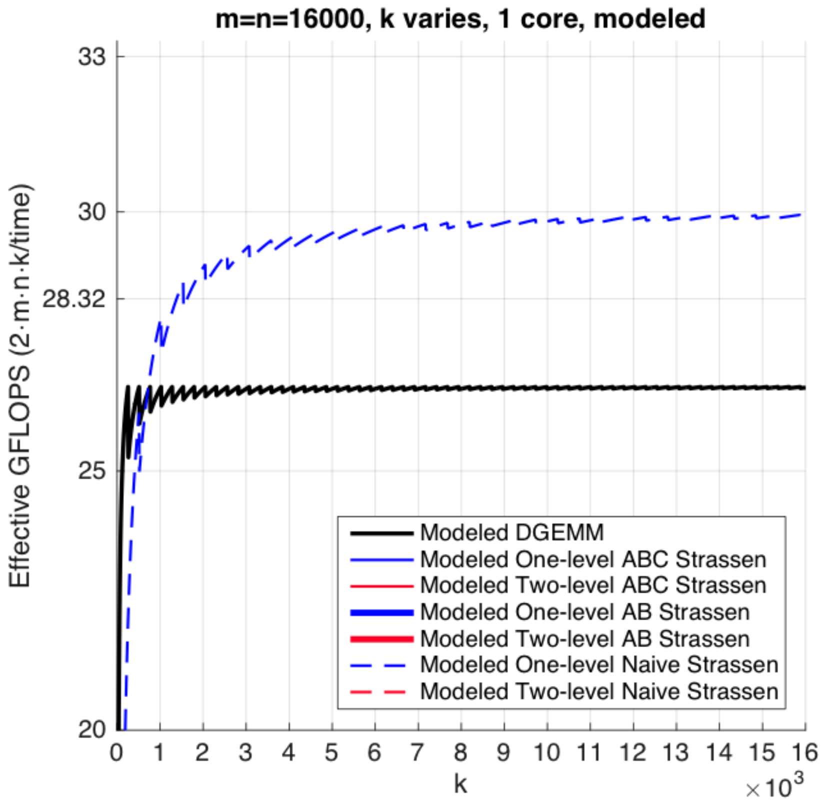


Actual Performance

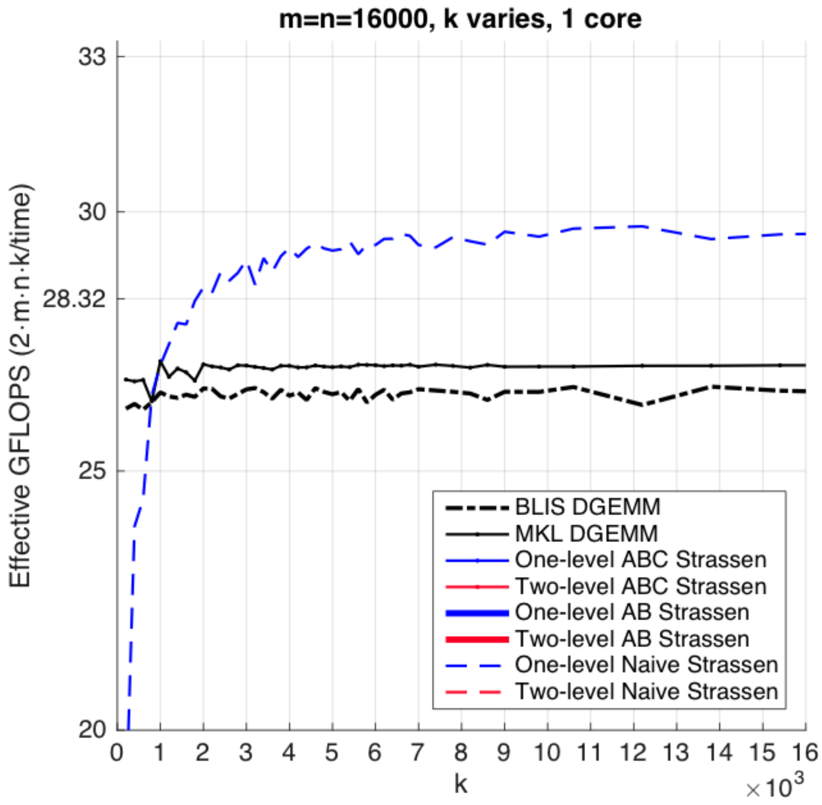


Observation (Rank-k Update)

Modeled Performance

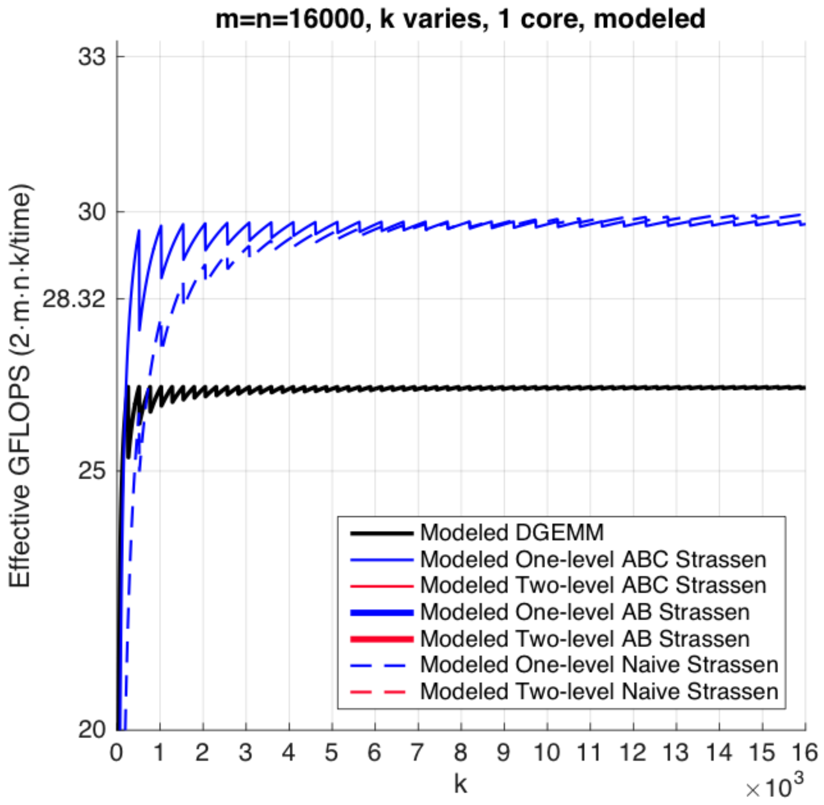


Actual Performance

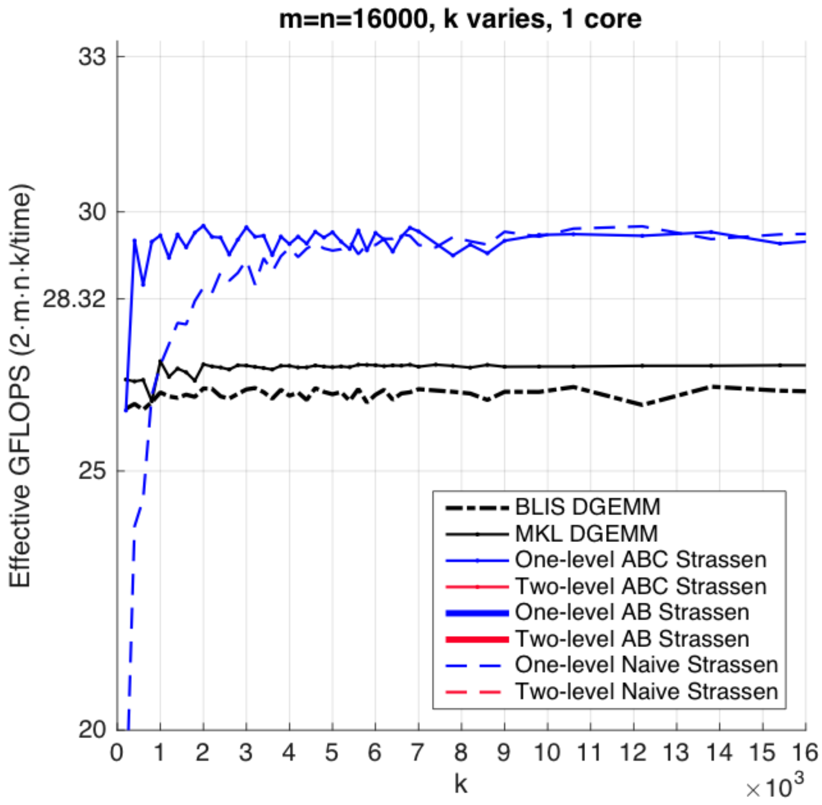


Observation (Rank-k Update)

Modeled Performance

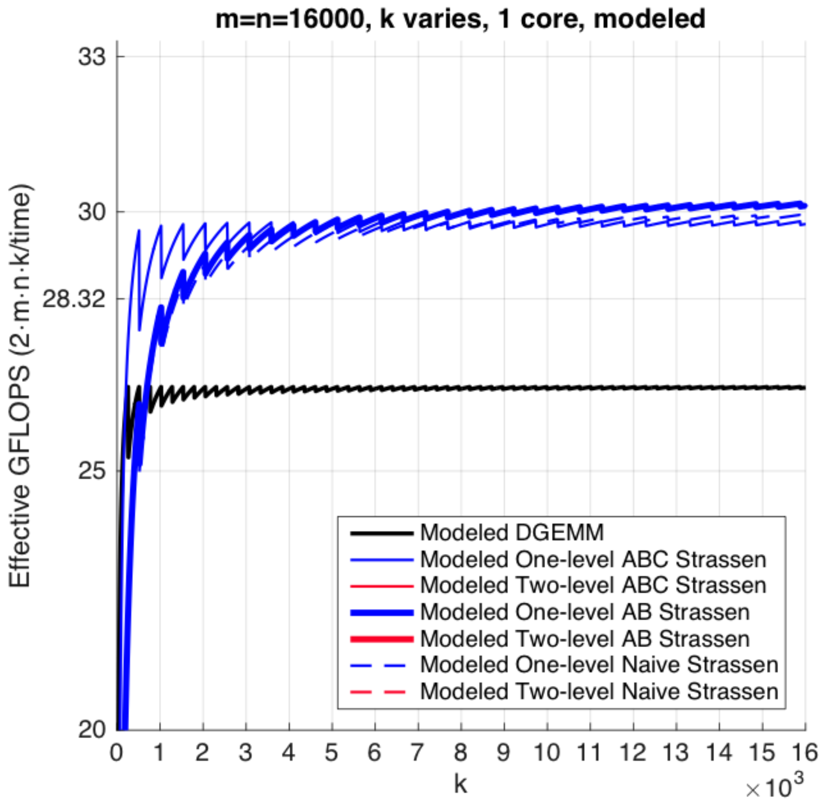


Actual Performance

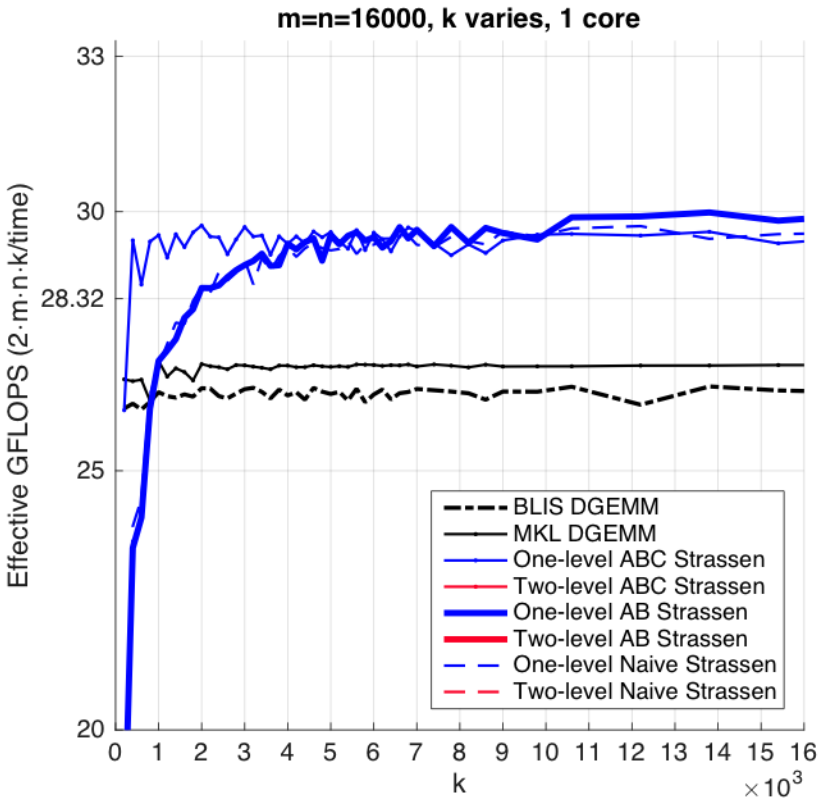


Observation (Rank-k Update)

Modeled Performance

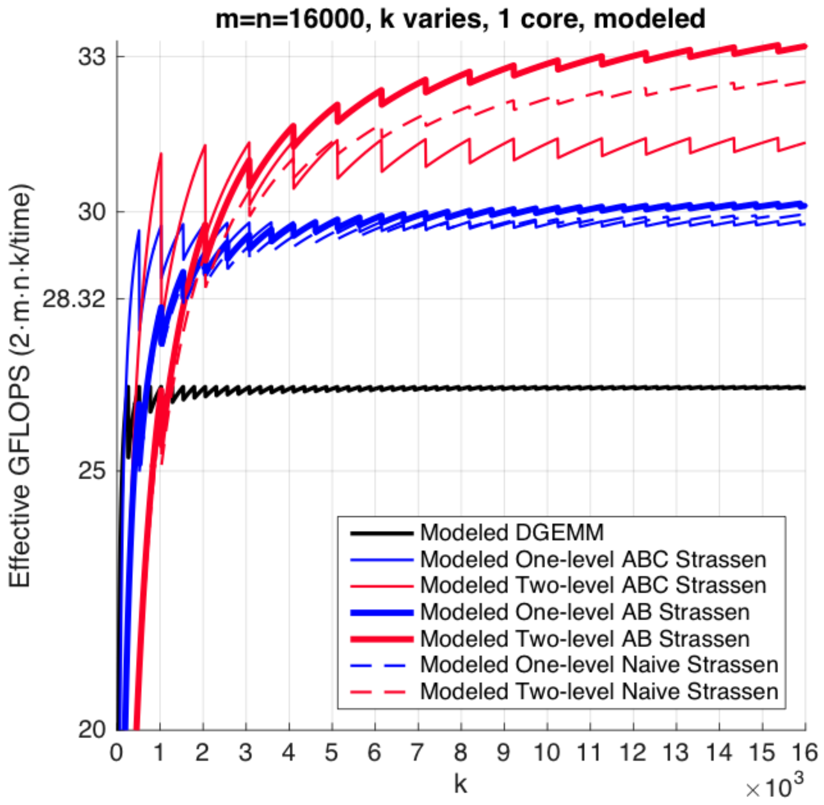


Actual Performance

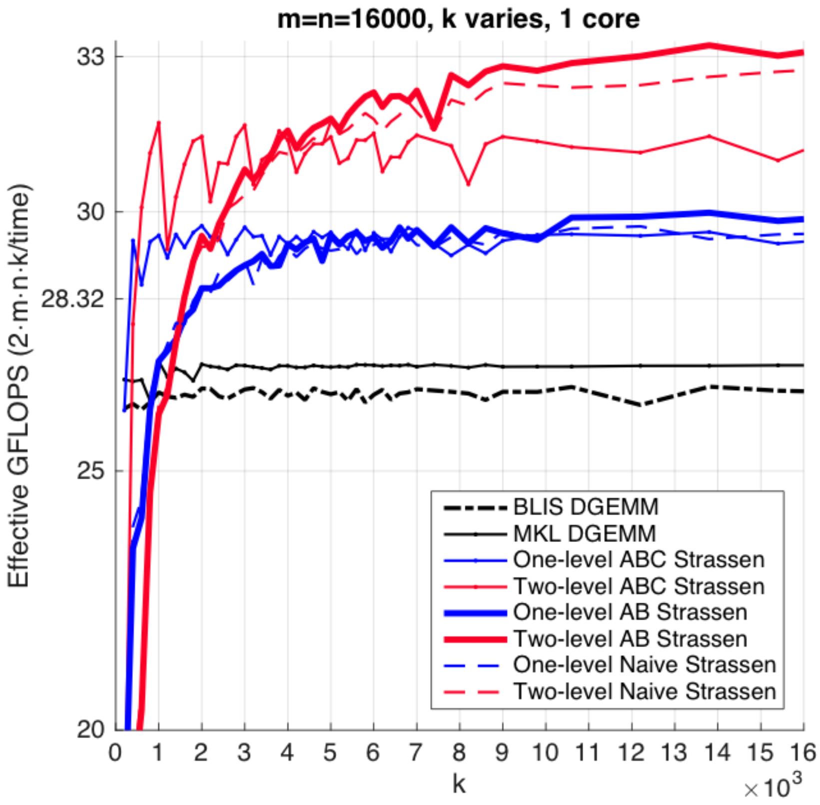


Observation (Rank-k Update)

Modeled Performance

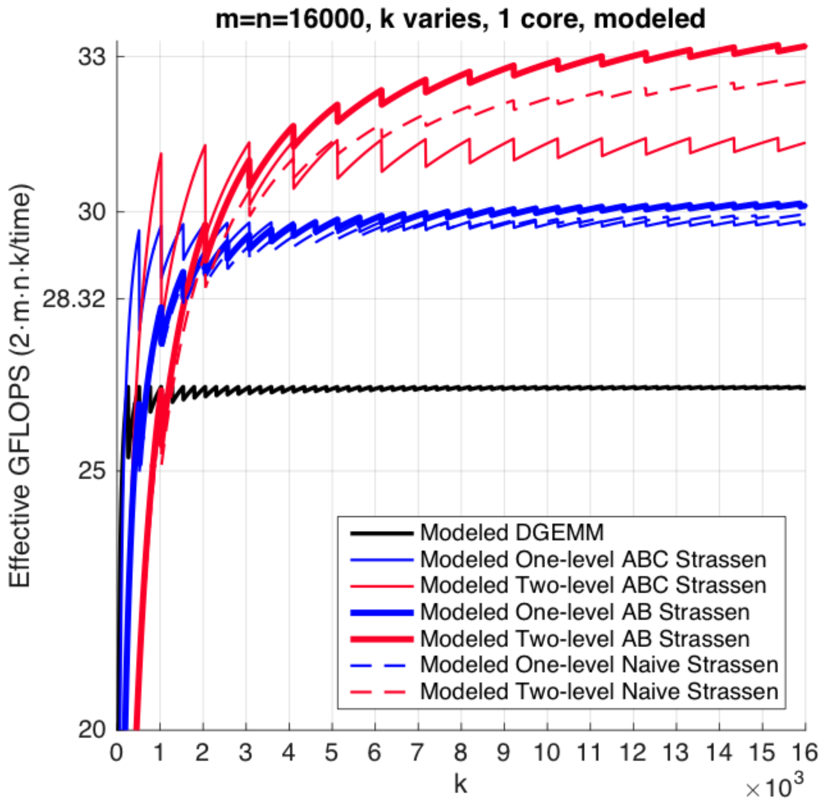


Actual Performance

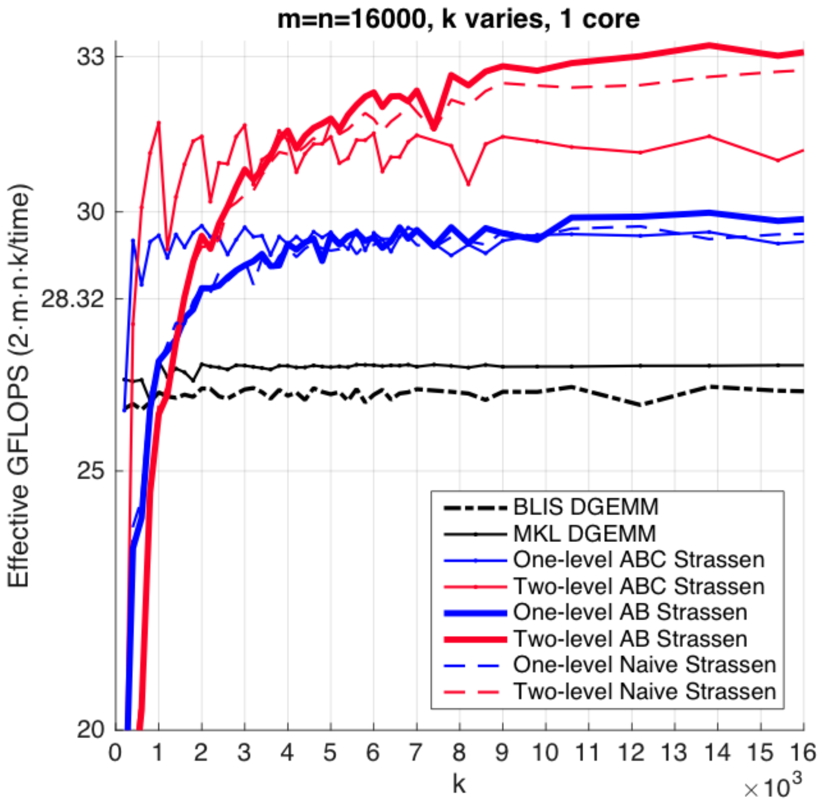


Observation (Rank-k Update)

Modeled Performance



Actual Performance



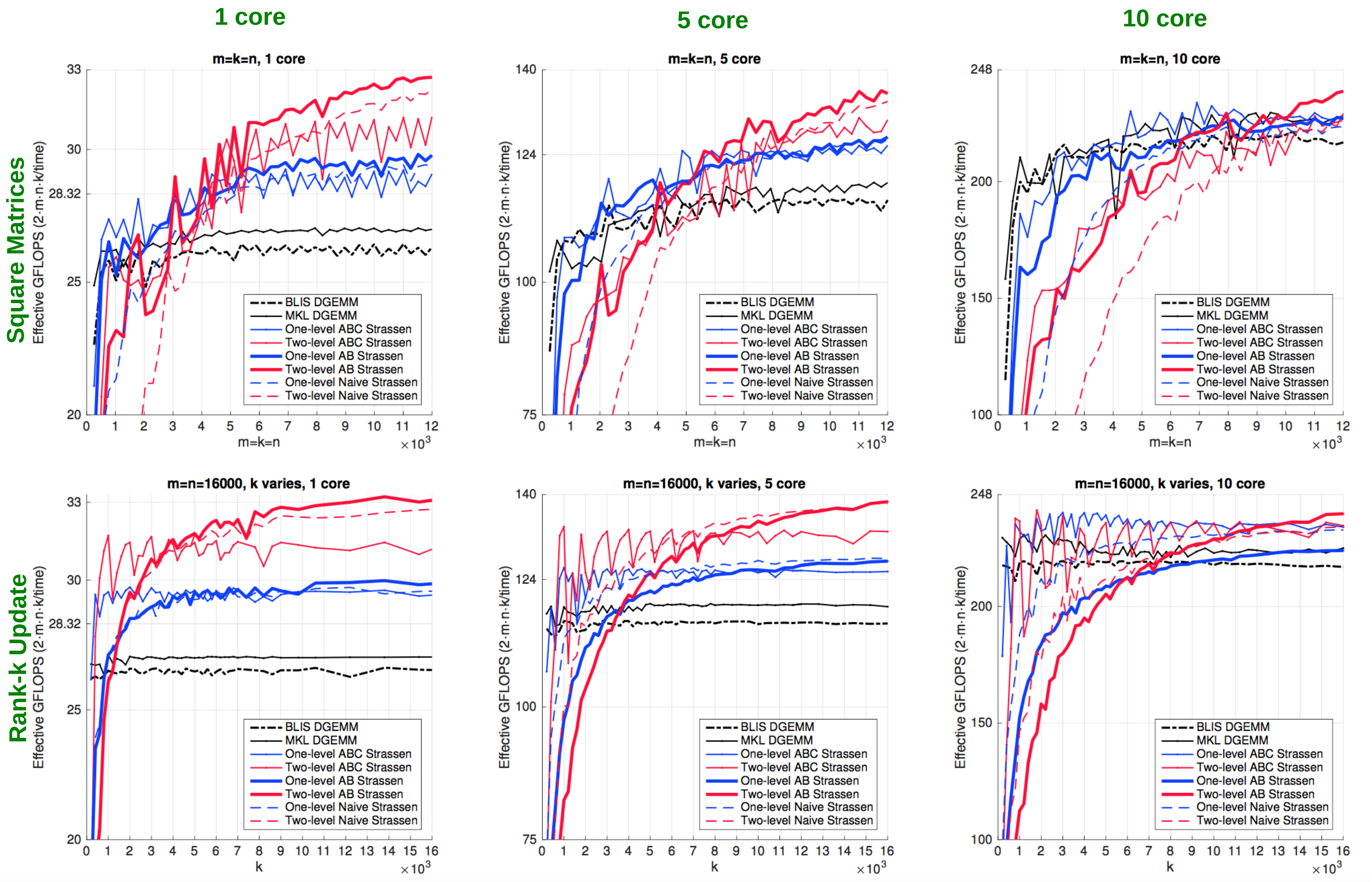
- Reason:

  ABC Strassen avoids forming the temporary matrix M explicitly in the memory (M resides in register), especially important when m, n >> k.

Outline

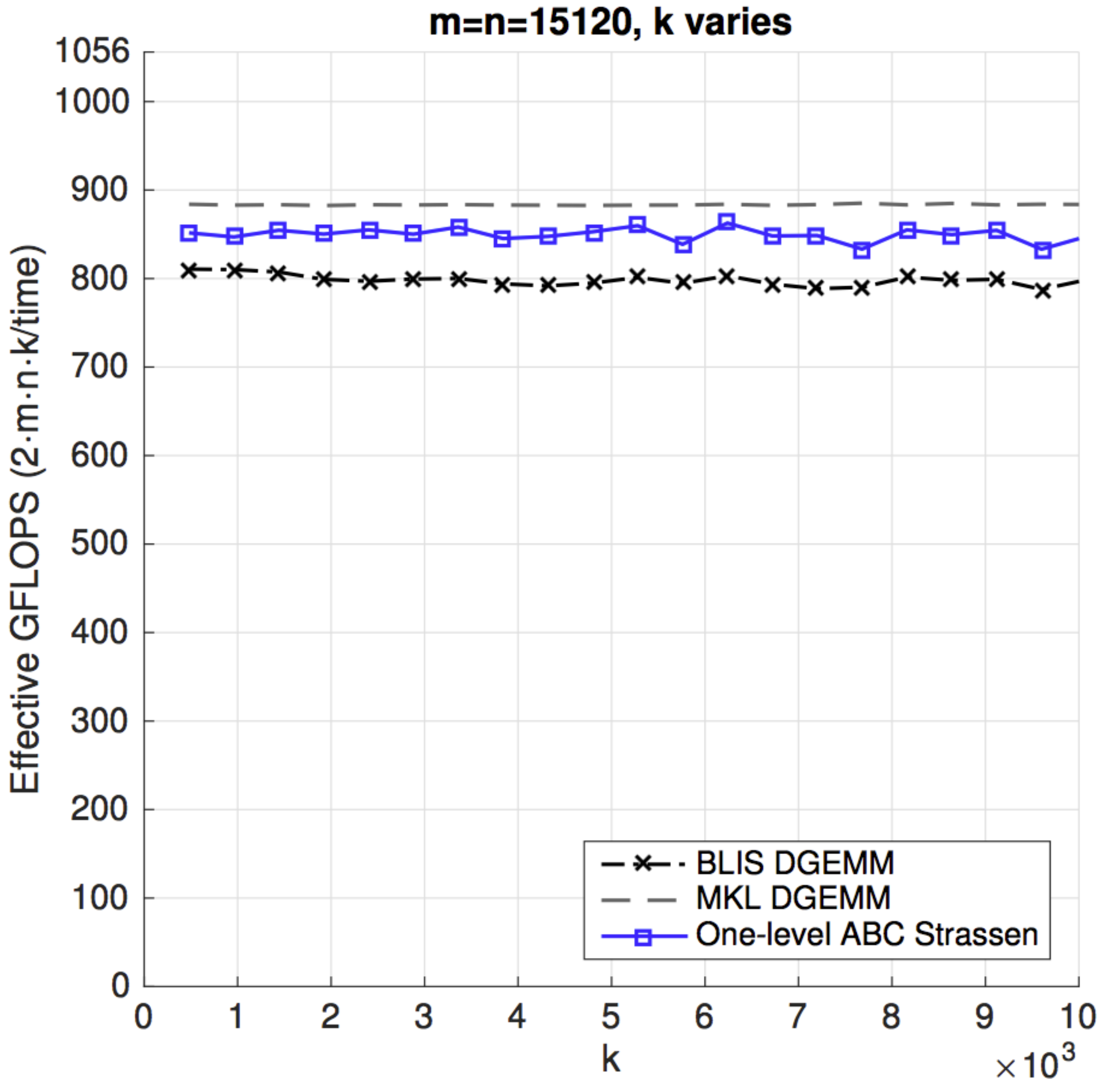
- Standard Matrix-matrix multiplication
- Strassen's Algorithm Reloaded
- Theoretical Model and Analysis
- Performance Experiments
- Conclusion

Single Node Experiment

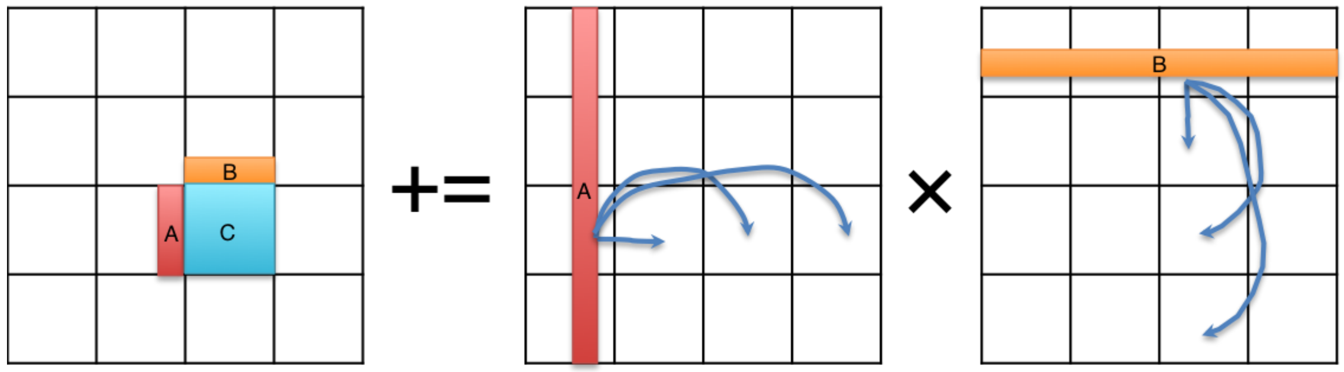

1 core
5 core
10 core
Square Matrices
Rank-k Update
m=k=n, 1 core
m=k=n, 5 core
m=k=n, 10 core
m=n=16000, k varies, 1 core
m=n=16000, k varies, 5 core
m=n=16000, k varies, 10 core
Effective GFLOPS (2·m·n·k/time)
m=k=n
k
×10^3
28.32
BLIS DGEMM
MKL DGEMM
One-level ABC Strassen
Two-level ABC Strassen
One-level AB Strassen
Two-level AB Strassen
One-level Naive Strassen
Two-level Naive Strassen

Many-core Experiment

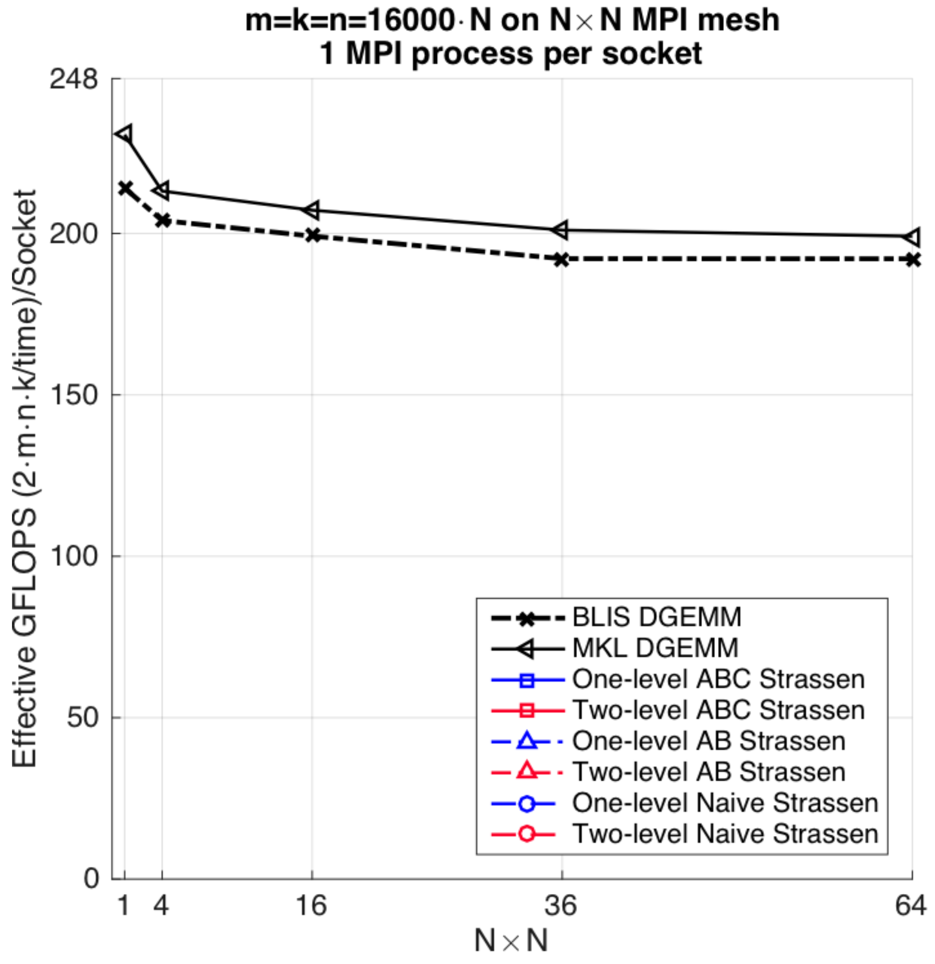
Intel® Xeon Phi™ coprocessor (KNC)



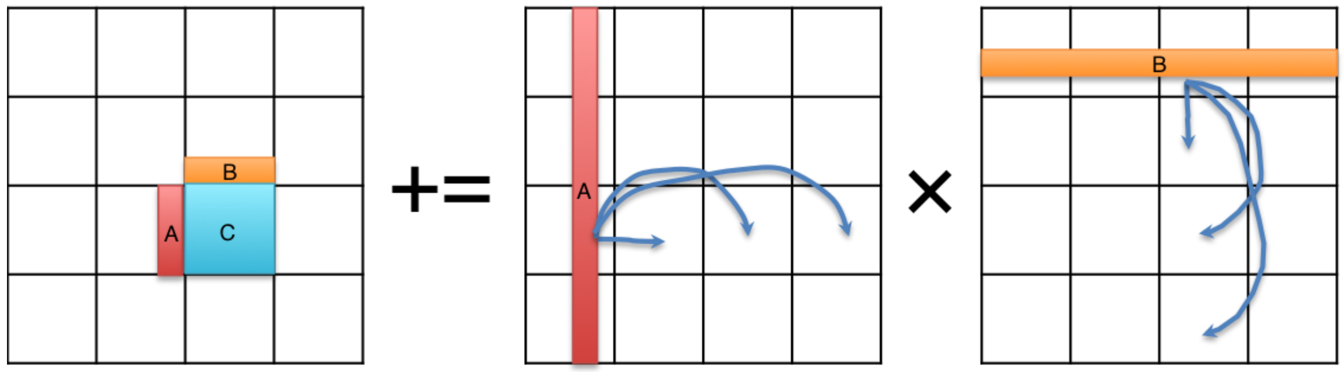
Distributed Memory Experiment



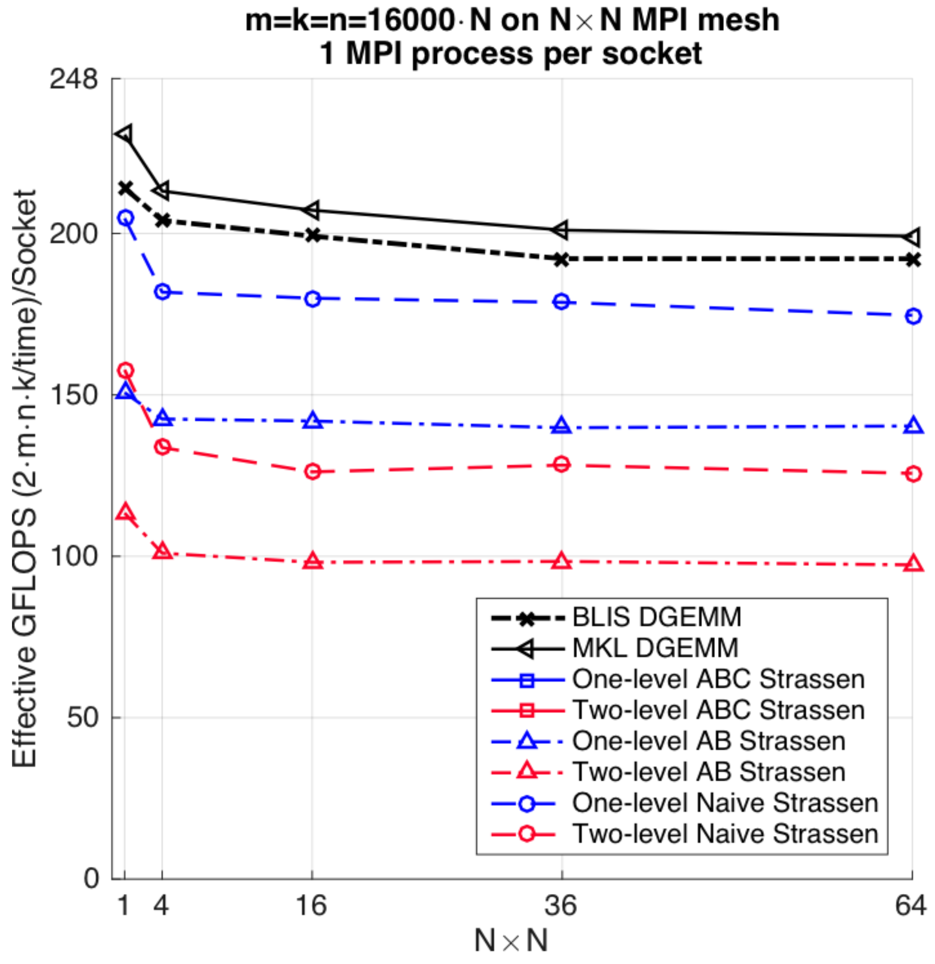
B
A
C
+=
A
×
B



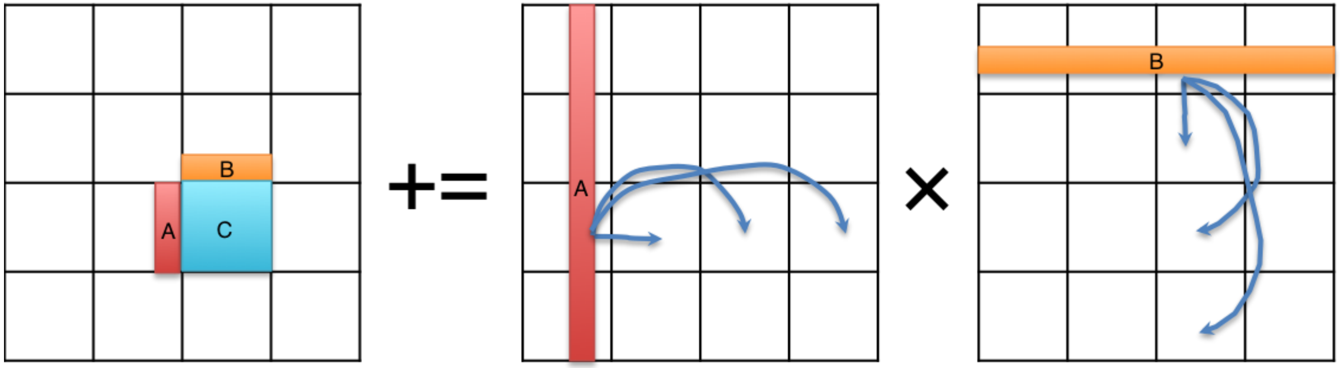
m=k=n=16000·N on N×N MPI mesh
1 MPI process per socket
Effective GFLOPS (2·m·n·k/time)/Socket
248
200
150
100
50
0
1
4
16
36
64
N×N
BLIS DGEMM
MKL DGEMM
One-level ABC Strassen
Two-level ABC Strassen
One-level AB Strassen
Two-level AB Strassen
One-level Naive Strassen
Two-level Naive Strassen

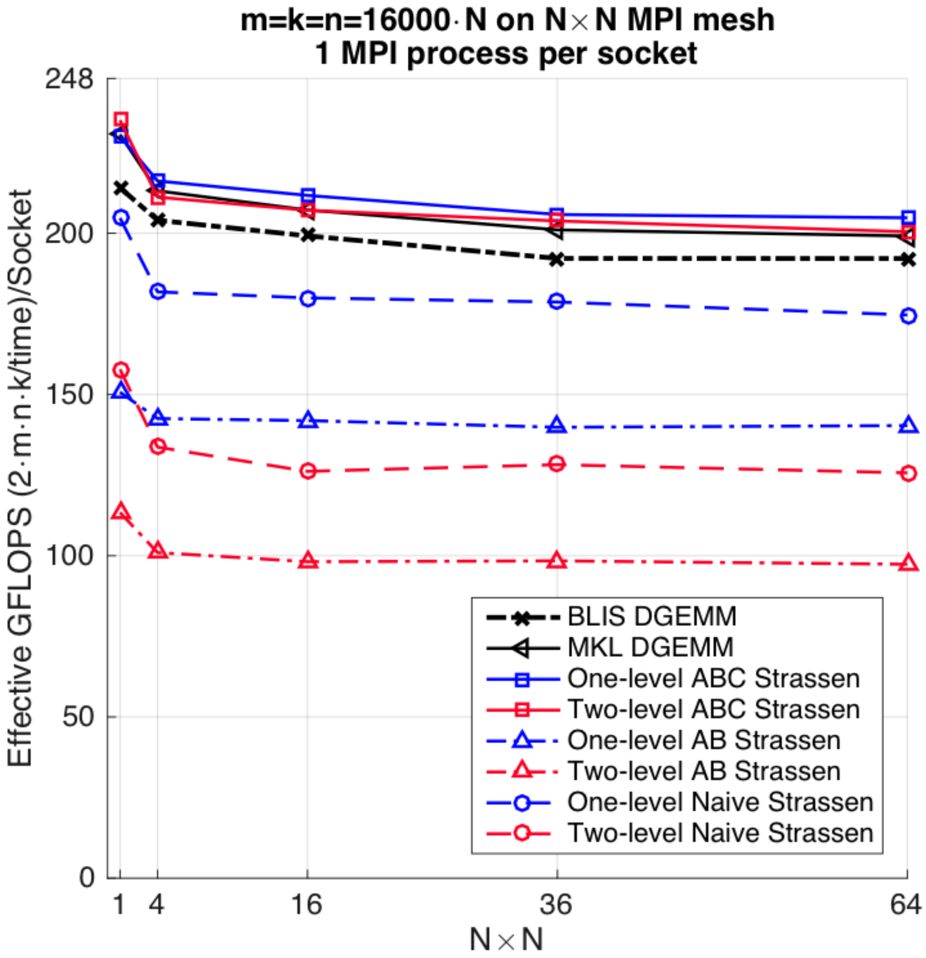


B
A
C
+=
A
×
B



m=k=n=16000·N on N×N MPI mesh
1 MPI process per socket
Effective GFLOPS (2·m·n·k/time)/Socket
248
200
150
100
50
0
1
4
16
36
64
N×N
BLIS DGEMM
MKL DGEMM
One-level ABC Strassen
Two-level ABC Strassen
One-level AB Strassen
Two-level AB Strassen
One-level Naive Strassen
Two-level Naive Strassen

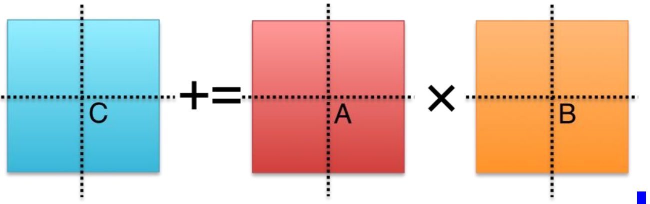

A
B
C
+=
A
×
B


m=k=n=16000·N on N×N MPI mesh
1 MPI process per socket
Effective GFLOPS (2·m·n·k/time)/Socket
248
200
150
100
50
0
1
4
16
36
64
N×N
BLIS DGEMM
MKL DGEMM
One-level ABC Strassen
Two-level ABC Strassen
One-level AB Strassen
Two-level AB Strassen
One-level Naive Strassen
Two-level Naive Strassen

Outline

- Standard Matrix-matrix multiplication
- Strassen’s Algorithm Reloaded
- Theoretical Model and Analysis
- Performance Experiments
- Conclusion

To achieve practical high performance of Strassen's algorithm……



	Conventional Implementations	Our Implementations
Matrix Size	**Must be large**	C += A × B
Matrix Shape	**Must be square**	C += A × B
No Additional Workspace	✗	✓
Parallelism	**Usually task parallelism**	**Can be data parallelism**

Acknowledgement

- NSF grants ACI-1148125/1340293, CCF-1218483.
- Intel Corporation through an Intel Parallel Computing Center (IPCC).
- Access to the Maverick and Stampede supercomputers administered by TACC.

We thank Field Van Zee, Chenhan Yu, Devin Matthews, and the rest of the SHPC team (http://shpc.ices.utexas.edu) for their supports.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Thank you!