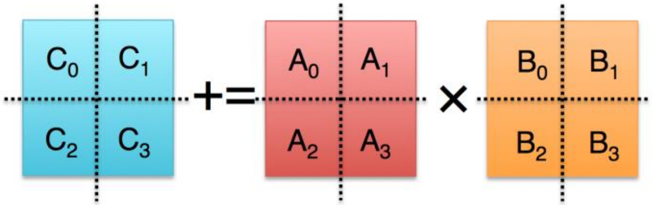
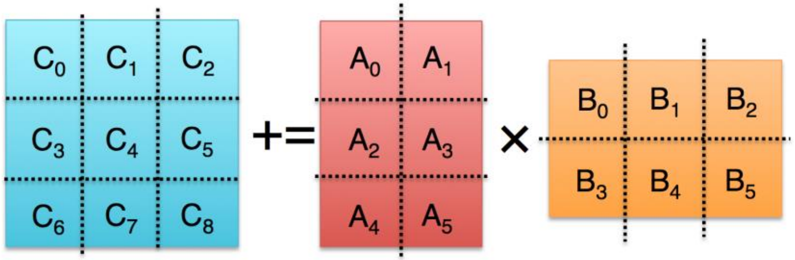




Generating Families of Practical Fast Matrix Multiplication Algorithms

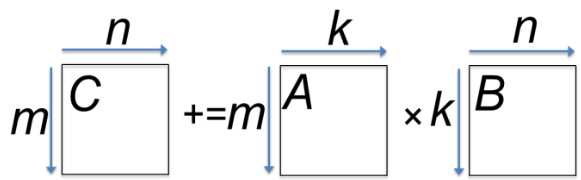
Jianyu Huang, Leslie Rice,
Devin A. Matthews, Robert A. van de Geijn
The University of Texas at Austin

Outline

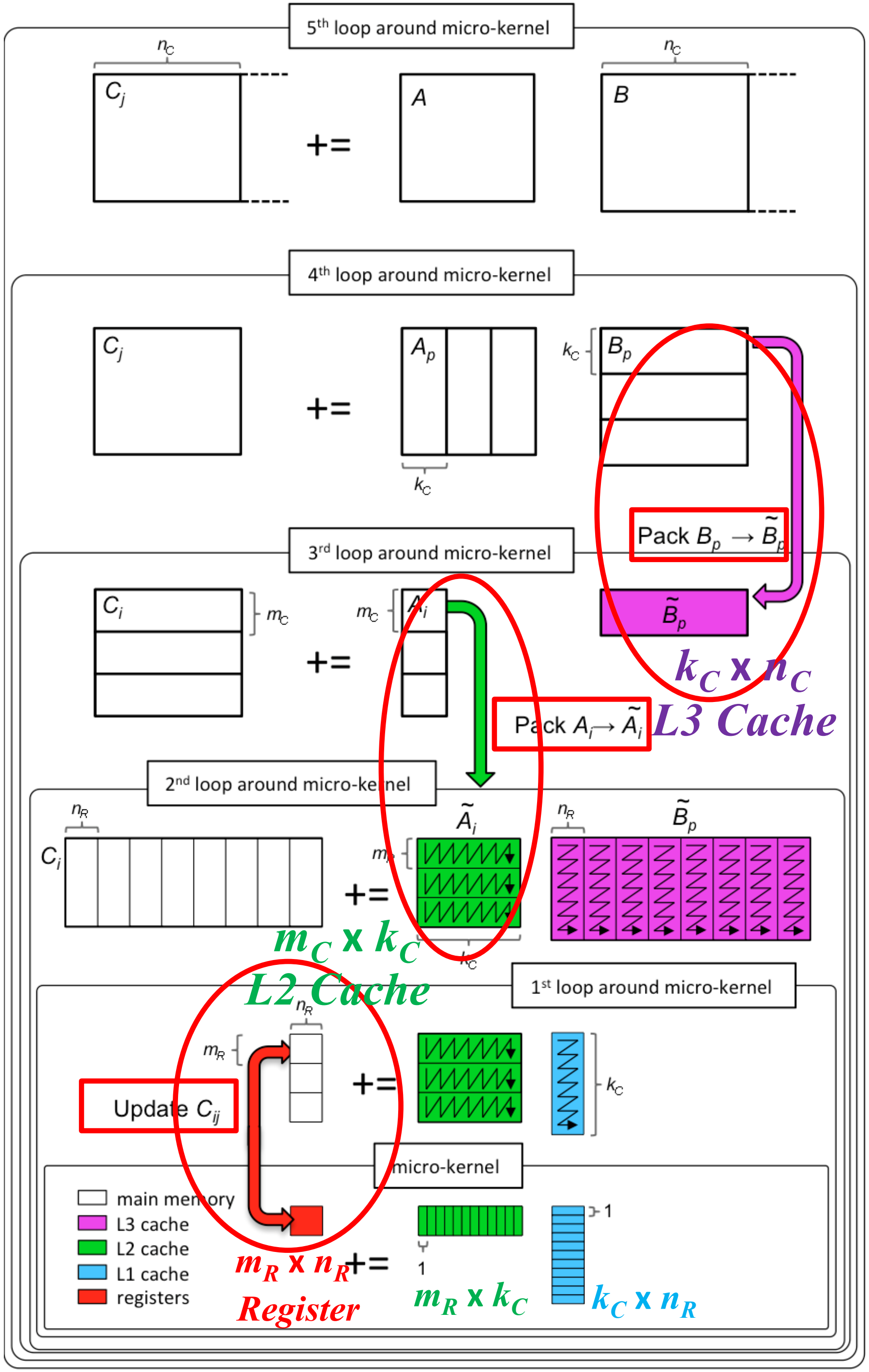
- Background
  - High-performance GEMM
  - High-performance Strassen
- Fast Matrix Multiplication (FMM)
- Code Generation
- Performance Model
- Experiments
- Conclusion

High-performance matrix multiplication (GEMM)

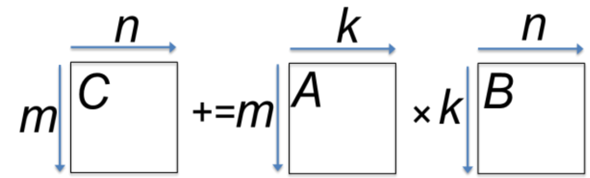
State-of-the-art GEMM in BLIS



- 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 on CPU) 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.



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

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

endfor
endfor
endfor
endfor
endfor
endfor

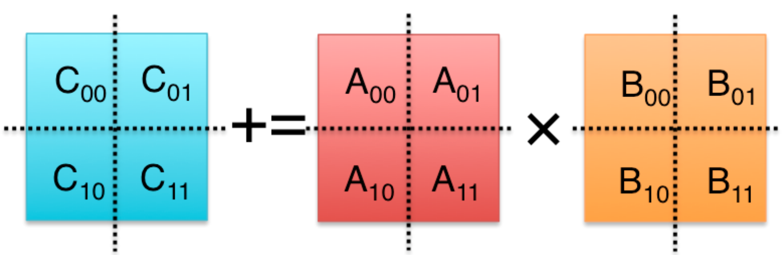
*Field G. Van Zee, and Tyler M. Smith. "Implementing high-performance complex matrix multiplication via the 3m and 4m methods." In *ACM Transactions on Mathematical Software (TOMS), accepted.*

High-performance Strassen

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

Strassen's Algorithm Reloaded

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



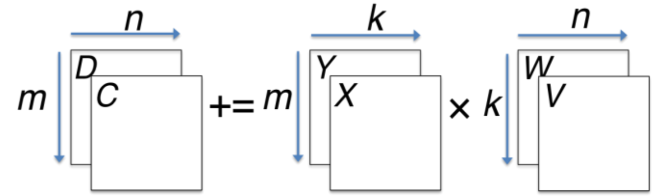
$\boxed{M_0 := (A_{00}+A_{11})(B_{00}+B_{11}); \quad C_{00} \mathrel{+}= M_0;\ C_{11} \mathrel{+}= M_0;}$
$M_1 := (A_{10}+A_{11})B_{00}; \quad C_{10} \mathrel{+}= M_1;\ C_{11} \mathrel{-}= M_1;$
$M_2 := A_{00}(B_{01}-B_{11}); \quad C_{01} \mathrel{+}= M_2;\ C_{11} \mathrel{+}= M_2;$
$M_3 := A_{11}(B_{10}-B_{00});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

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

General operation for one-level Strassen:

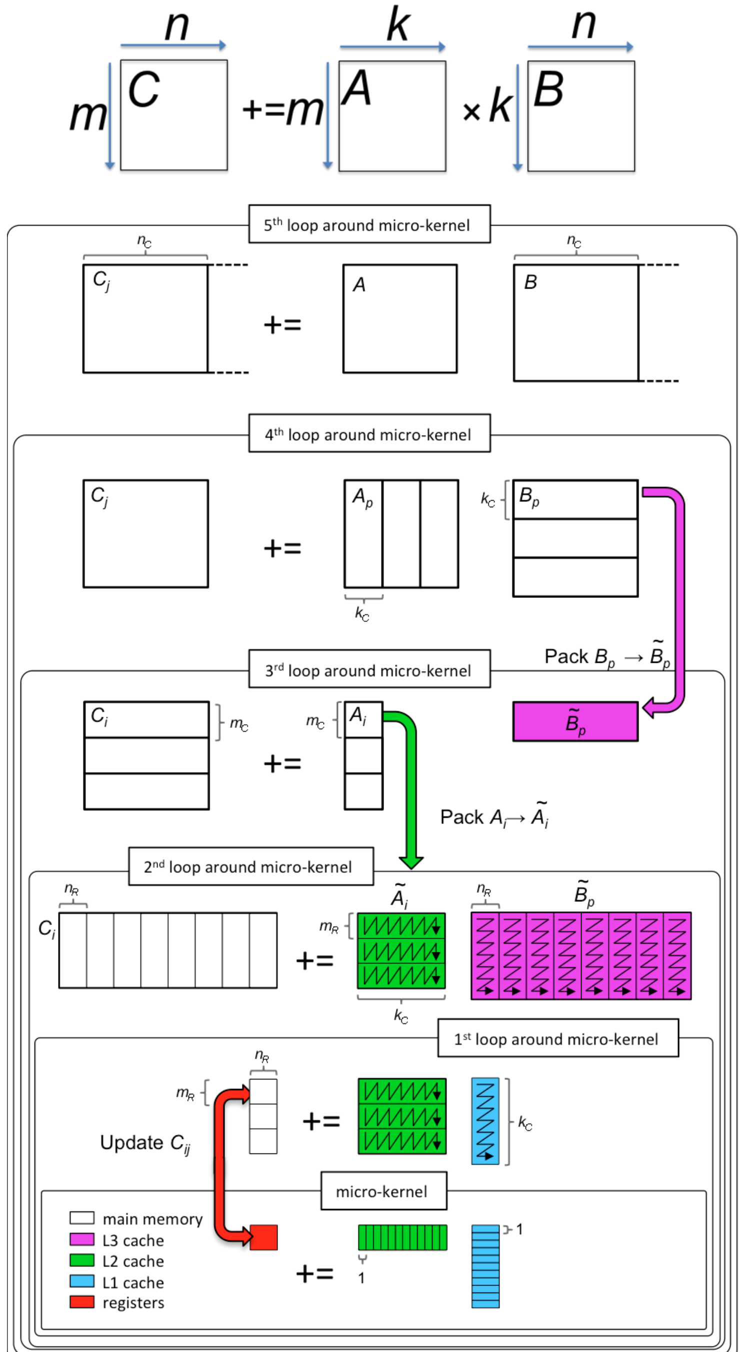
$\boxed{M := (X+\delta Y)(V+\varepsilon W); \quad C \mathrel{+}= \gamma_0 M;\ D \mathrel{+}= \gamma_1 M; \quad \gamma_0, \gamma_1, \delta, \varepsilon \in \{-1, 0, 1\}.}$

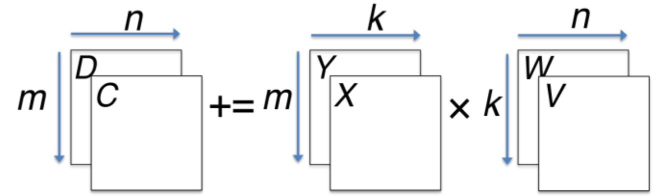
$$M := (X + Y)(V + W); \quad C += M; \; D += M;$$

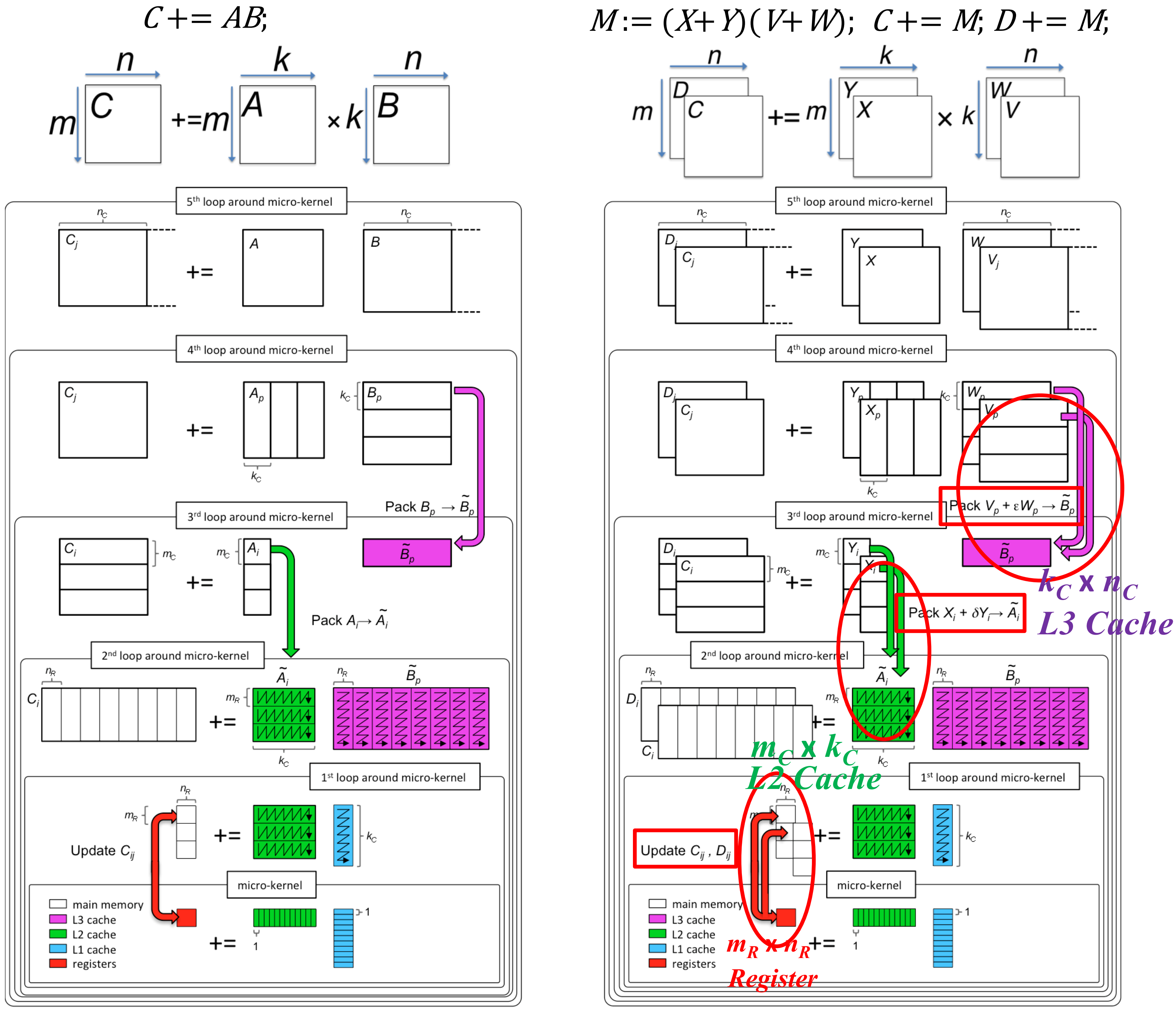


$$C += AB;$$

$$M := (X + Y)(V + W);\; C += M;\; D += M;$$







$C += AB;$
$M := (X+Y)(V+W); \; C += M; \; D += M;$
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 $A_i \rightarrow \tilde{A}_i$
Pack $V_p + \varepsilon W_p \rightarrow \tilde{B}_p$
Pack $X_i + \delta Y_i \rightarrow \tilde{A}_i$
Update C_{ij}
Update C_{ij} , D_{ij}
main memory
L3 cache
L2 cache
L1 cache
registers
k_C x n_C
L3 Cache
m_C x k_C
L2 Cache
m_R x n_R
Register

Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
- Fast Matrix Multiplication (FMM)
- Code Generation
- Performance Model
- Experiments
- Conclusion

<2,2,2> Strassen's Algorithm

$M_0 := (A_0+A_3)(B_0+B_3);\quad C_0 \mathrel{+}= M_0;\ C_3 \mathrel{+}= M_0;$

$M_1 := (A_2+A_3)B_0;\quad C_2 \mathrel{+}= M_1;\ C_3 \mathrel{-}= M_1;$

$M_2 := A_0(B_1-B_3);\quad C_1 \mathrel{+}= M_2;\ C_3 \mathrel{+}= M_2;$

$M_3 := A_3(B_2-B_0);\quad C_0 \mathrel{+}= M_3;\ C_2 \mathrel{+}= M_3;$

$M_4 := (A_0+A_1)B_3;\quad C_1 \mathrel{+}= M_4;\ C_0 \mathrel{-}= M_4;$

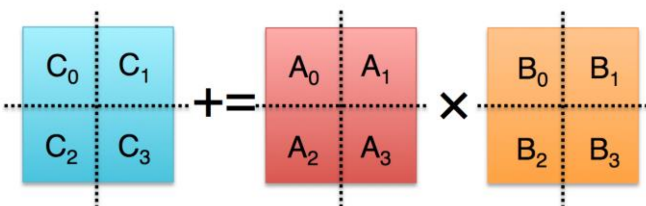
$M_5 := (A_2-A_0)(B_0+B_1);\quad C_3 \mathrel{+}= M_5;$

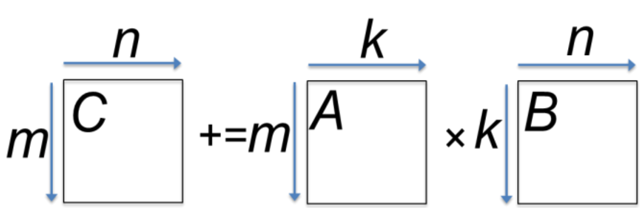
$M_6 := (A_1-A_3)(B_2+B_3);\quad C_0 \mathrel{+}= M_6;$

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

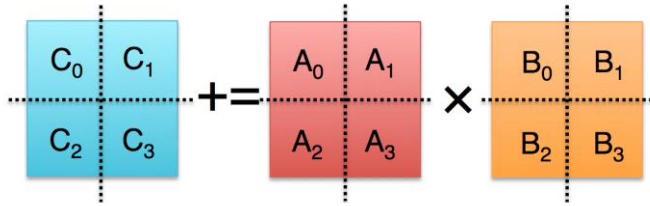
$$\mathbf{V} = \begin{bmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$





$\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$ Fast Matrix Multiplication (FMM)



$$C = \left(\begin{array}{c|c|c} C_0 & \cdots & C_{\widetilde{n}-1} \\ \hline \vdots & & \vdots \\ \hline C_{(\widetilde{m}-1)\widetilde{n}} & \cdots & C_{\widetilde{m}\widetilde{n}-1} \end{array} \right), A = \left(\begin{array}{c|c|c} A_0 & \cdots & A_{\widetilde{k}-1} \\ \hline \vdots & & \vdots \\ \hline A_{(\widetilde{m}-1)\widetilde{k}} & \cdots & A_{\widetilde{m}\widetilde{k}-1} \end{array} \right), B = \left(\begin{array}{c|c|c} B_0 & \cdots & B_{\widetilde{n}-1} \\ \hline \vdots & & \vdots \\ \hline B_{(\widetilde{k}-1)\widetilde{n}} & \cdots & B_{\widetilde{k}\widetilde{n}-1} \end{array} \right)$$

$$\text{for} \quad r = 0, \ldots, R-1,$$

$$M_r := \left(\sum_{i=0}^{\widetilde{m}\widetilde{k}-1} u_{ir} A_i \right) \times \left(\sum_{j=0}^{\widetilde{k}\widetilde{n}-1} v_{jr} B_j \right);$$

$$C_p += w_{pr} M_r \ (p = 0, \ldots, \widetilde{m}\widetilde{n} - 1)$$

The set of coefficients that determine the $\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$ algorithm is denoted as $[\![U, V, W]\!]$.

<3,2,3> Fast Matrix Multiplication (FMM)

$M_0 := (-A_1+A_3+A_5)(B_4+B_5);$ $\quad C_2 -= M_0;$

$M_1 := (-A_0+A_2+A_5)(B_0-B_5);$ $\quad C_2 -= M_1;\ C_7 -= M_1;$

$M_2 := (-A_0+A_2)(-B_1-B_4);$ $\quad C_0-=M_2;C_1+=M_2;C_7+=M_2;$

$M_3 := (A_2-A_4+A_5)(B_2-B_3+B_5);$ $\quad C_2+=M_3;C_5+=M_3;C_6-=M_3;$

$M_4 := (-A_0+A_1+A_2)(B_0+B_1+B_4);$ $\quad C_0-=M_4;C_1+=M_4;C_4+=M_4;$

$M_5 := (A_2)(B_0);$ $\quad C_0+=M_5;C_1-=M_5;C_3+=M_5;C_4-=M_5;C_6+=M_5;$

$M_6 := (-A_2+A_3+A_4-A_5)(B_3-B_5);$ $\quad C_2-=M_6;C_5-=M_6;$

$M_7 := (A_3)(B_3);$ $\quad C_2+=M_7;C_3+=M_7;C_5+=M_7;$

$M_8 := (-A_0+A_2+A_4)(B_1+B_2);$ $\quad C_7+=M_8;$

$M_9 := (-A_0+A_1)(-B_0-B_1);$ $\quad C_1+=M_9;C_4+=M_9;$

$M_{10} :=(A_5)(B_2+B_5);$ $\quad C_2+=M_{10};C_6+=M_{10};C_8+=M_{10};$

$M_{11} := (A_4-A_5)(B_2);$ $\quad C_2+=M_{11};C_5+=M_{11};C_7-=M_{11};C_8+=M_{11}$

$M_{12} := (A_1)(B_0+B_1+B_3+B_4);$ $\quad C_0+=M_{12};$

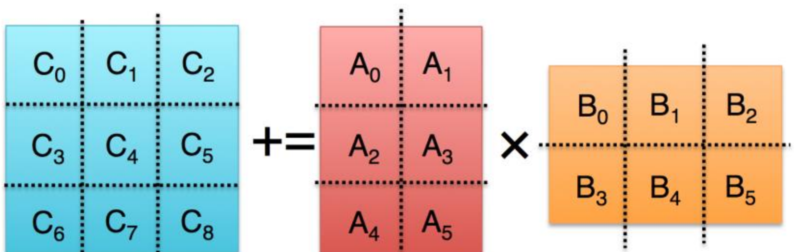
$M_{13} := (A_2-A_4)(B_0-B_2+B_3-B_5);$ $\quad C_6-=M_{13};$

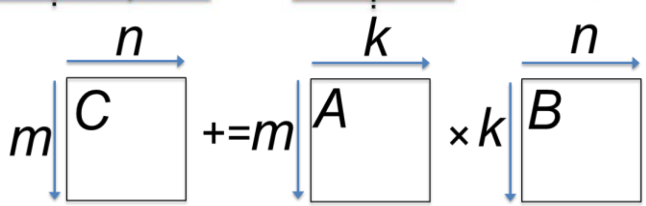
$M_{14} := (-A_0+A_1+A_2-A_3)(B_5);$ $\quad C_2-=M_{14};C_4-=M_{14};$

$$\mathbf{U} = \begin{bmatrix} 0 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$





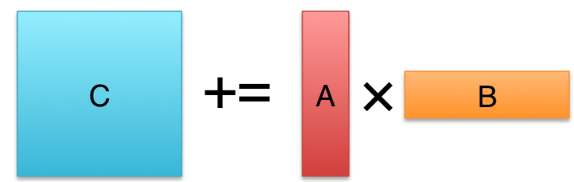
*Austin R. Benson, Grey Ballard. "A framework for practical parallel fast matrix multiplication." *PPoPP*15.

$\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$	Ref. $\widetilde{m}\widetilde{k}\widetilde{n}$ R	Speedup (%)				
		Theory	Practical #1		Practical #2	
			Ours	[1]	Ours	[1]
$\langle 2, 2, 2 \rangle$						
$\langle 2, 3, 2 \rangle$						
$\langle 2, 3, 4 \rangle$						
$\langle 2, 4, 3 \rangle$						
$\langle 2, 5, 2 \rangle$						
$\langle 3, 2, 2 \rangle$						
$\langle 3, 2, 3 \rangle$						
$\langle 3, 2, 4 \rangle$						
$\langle 3, 3, 2 \rangle$						
$\langle 3, 3, 3 \rangle$						
$\langle 3, 3, 6 \rangle$						
$\langle 3, 4, 2 \rangle$						
$\langle 3, 4, 3 \rangle$						
$\langle 3, 5, 3 \rangle$						
$\langle 3, 6, 3 \rangle$						
$\langle 4, 2, 2 \rangle$						
$\langle 4, 2, 3 \rangle$						
$\langle 4, 2, 4 \rangle$						
$\langle 4, 3, 2 \rangle$						
$\langle 4, 3, 3 \rangle$						
$\langle 4, 4, 2 \rangle$						
$\langle 5, 2, 2 \rangle$						
$\langle 6, 3, 3 \rangle$						

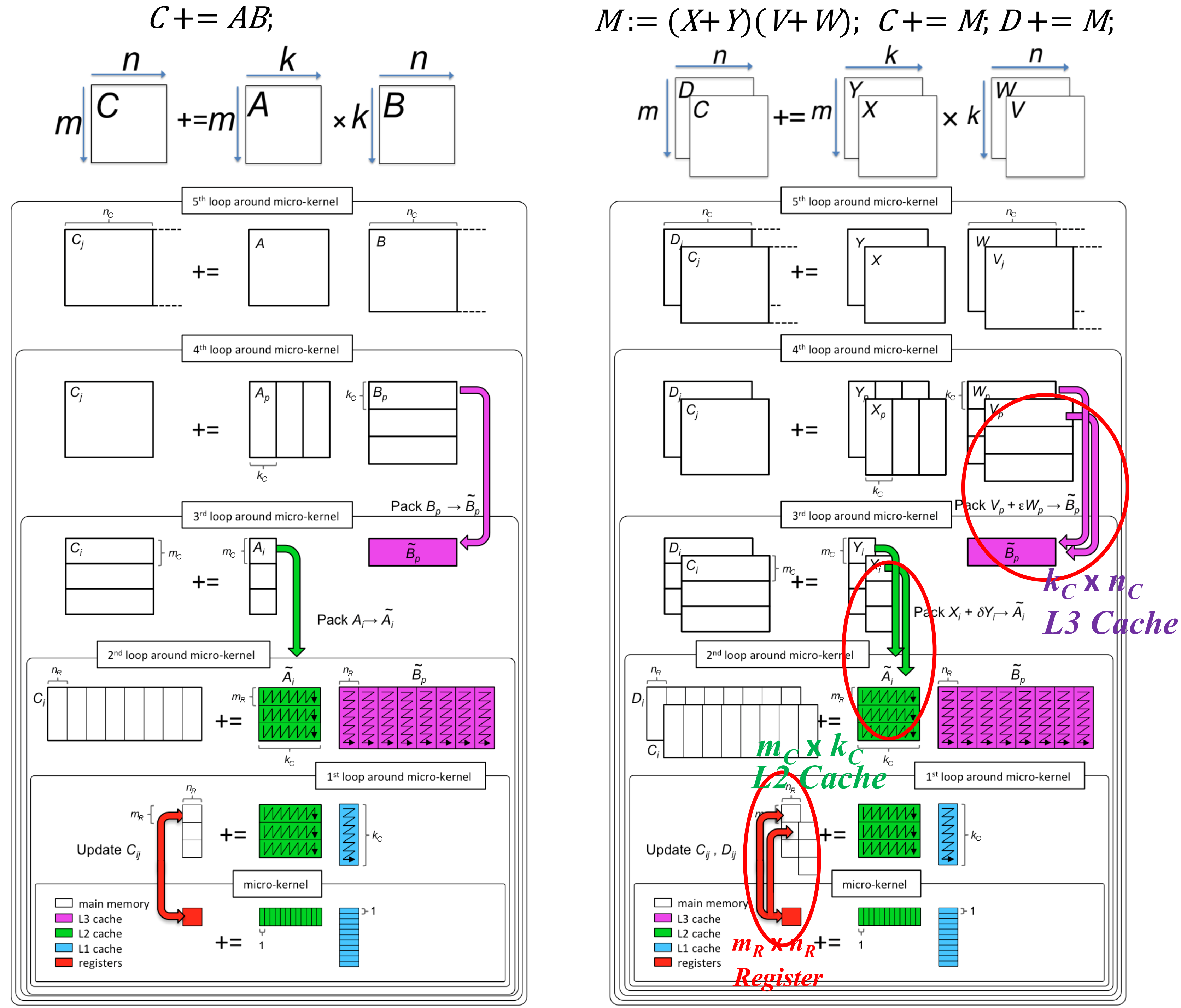
$\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$	Ref.	$\widetilde{m}\widetilde{k}\widetilde{n}$	R	Speedup (%)				
				Theory	Practical #1		Practical #2	
					Ours	[1]	Ours	[1]
$\langle 2,2,2 \rangle$	[13]	8	7					
$\langle 2,3,2 \rangle$	[1]	12	11					
$\langle 2,3,4 \rangle$	[1]	24	20					
$\langle 2,4,3 \rangle$	[10]	24	20					
$\langle 2,5,2 \rangle$	[10]	20	18					
$\langle 3,2,2 \rangle$	[10]	12	11					
$\langle 3,2,3 \rangle$	[10]	18	15					
$\langle 3,2,4 \rangle$	[10]	24	20					
$\langle 3,3,2 \rangle$	[10]	18	15					
$\langle 3,3,3 \rangle$	[14]	27	23					
$\langle 3,3,6 \rangle$	[14]	54	40					
$\langle 3,4,2 \rangle$	[1]	24	20					
$\langle 3,4,3 \rangle$	[14]	36	29					
$\langle 3,5,3 \rangle$	[14]	45	36					
$\langle 3,6,3 \rangle$	[14]	54	40					
$\langle 4,2,2 \rangle$	[10]	16	14					
$\langle 4,2,3 \rangle$	[1]	24	20					
$\langle 4,2,4 \rangle$	[10]	32	26					
$\langle 4,3,2 \rangle$	[10]	24	20					
$\langle 4,3,3 \rangle$	[10]	36	29					
$\langle 4,4,2 \rangle$	[10]	32	26					
$\langle 5,2,2 \rangle$	[10]	20	18					
$\langle 6,3,3 \rangle$	[14]	54	40					

$\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$	Ref.	$\widetilde{m}\widetilde{k}\widetilde{n}$	R	Speedup (%)				
				Theory	Practical #1		Practical #2	
					Ours	[1]	Ours	[1]
$\langle 2,2,2 \rangle$	[13]	8	7	14.3				
$\langle 2,3,2 \rangle$	[1]	12	11	9.1				
$\langle 2,3,4 \rangle$	[1]	24	20	20.0				
$\langle 2,4,3 \rangle$	[10]	24	20	20.0				
$\langle 2,5,2 \rangle$	[10]	20	18	11.1				
$\langle 3,2,2 \rangle$	[10]	12	11	9.1				
$\langle 3,2,3 \rangle$	[10]	18	15	20.0				
$\langle 3,2,4 \rangle$	[10]	24	20	20.0				
$\langle 3,3,2 \rangle$	[10]	18	15	20.0				
$\langle 3,3,3 \rangle$	[14]	27	23	17.4				
$\langle 3,3,6 \rangle$	[14]	54	40	35.0				
$\langle 3,4,2 \rangle$	[1]	24	20	20.0				
$\langle 3,4,3 \rangle$	[14]	36	29	24.1				
$\langle 3,5,3 \rangle$	[14]	45	36	25.0				
$\langle 3,6,3 \rangle$	[14]	54	40	35.0				
$\langle 4,2,2 \rangle$	[10]	16	14	14.3				
$\langle 4,2,3 \rangle$	[1]	24	20	20.0				
$\langle 4,2,4 \rangle$	[10]	32	26	23.1				
$\langle 4,3,2 \rangle$	[10]	24	20	20.0				
$\langle 4,3,3 \rangle$	[10]	36	29	24.1				
$\langle 4,4,2 \rangle$	[10]	32	26	23.1				
$\langle 5,2,2 \rangle$	[10]	20	18	11.1				
$\langle 6,3,3 \rangle$	[14]	54	40	35.0				

$\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$	Ref.	$\widetilde{m}\widetilde{k}\widetilde{n}$	R	Speedup (%)				
				Theory	Practical #1		Practical #2	
					Ours	[1]	Ours	[1]
$\langle 2,2,2 \rangle$	[13]	8	7	14.3	11.9	-3.0		
$\langle 2,3,2 \rangle$	[1]	12	11	9.1	5.5	-13.1		
$\langle 2,3,4 \rangle$	[1]	24	20	20.0	11.9	-8.0		
$\langle 2,4,3 \rangle$	[10]	24	20	20.0	4.8	-15.3		
$\langle 2,5,2 \rangle$	[10]	20	18	11.1	1.5	-23.1		
$\langle 3,2,2 \rangle$	[10]	12	11	9.1	7.1	-6.6		
$\langle 3,2,3 \rangle$	[10]	18	15	20.0	14.1	-0.7		
$\langle 3,2,4 \rangle$	[10]	24	20	20.0	11.9	-1.8		
$\langle 3,3,2 \rangle$	[10]	18	15	20.0	11.4	-8.1		
$\langle 3,3,3 \rangle$	[14]	27	23	17.4	8.6	-9.3		
$\langle 3,3,6 \rangle$	[14]	54	40	35.0	-34.0	-41.6		
$\langle 3,4,2 \rangle$	[1]	24	20	20.0	4.9	-15.7		
$\langle 3,4,3 \rangle$	[14]	36	29	24.1	8.4	-12.6		
$\langle 3,5,3 \rangle$	[14]	45	36	25.0	5.2	-20.6		
$\langle 3,6,3 \rangle$	[14]	54	40	35.0	-21.6	-64.5		
$\langle 4,2,2 \rangle$	[10]	16	14	14.3	9.4	-4.7		
$\langle 4,2,3 \rangle$	[1]	24	20	20.0	12.1	-2.3		
$\langle 4,2,4 \rangle$	[10]	32	26	23.1	10.4	-2.7		
$\langle 4,3,2 \rangle$	[10]	24	20	20.0	11.3	-7.8		
$\langle 4,3,3 \rangle$	[10]	36	29	24.1	8.1	-8.4		
$\langle 4,4,2 \rangle$	[10]	32	26	23.1	-4.2	-18.4		
$\langle 5,2,2 \rangle$	[10]	20	18	11.1	7.0	-6.7		
$\langle 6,3,3 \rangle$	[14]	54	40	35.0	-33.4	-42.2		

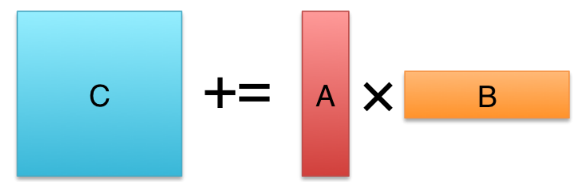


* [1] Austin R. Benson, Grey Ballard. "A framework for practical parallel fast matrix multiplication." *PPoPP*15.



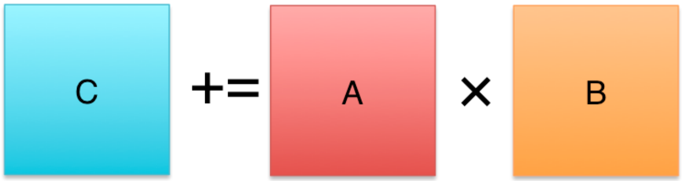
C += AB;
M := (X+Y)(V+W); C += M; D += M;
5th loop around micro-kernel
4th loop around micro-kernel
3rd loop around micro-kernel
2nd loop around micro-kernel
1st loop around micro-kernel
micro-kernel
Pack B_p → B̃_p
Pack A_i → Ã_i
Pack V_p + εW_p → B̃_p
Pack X_i + δY_i → Ã_i
Update C_ij
Update C_ij , D_ij
main memory
L3 cache
L2 cache
L1 cache
registers
k_C x n_C
L3 Cache
m_C x k_C
L2 Cache
m_R x n_R
Register

$\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$	Ref.	$\widetilde{m}\widetilde{k}\widetilde{n}$	R	Speedup (%)				
				Theory	Practical #1		Practical #2	
					Ours	[1]	Ours	[1]
$\langle 2,2,2 \rangle$	[13]	8	7	14.3	11.9	-3.0		
$\langle 2,3,2 \rangle$	[1]	12	11	9.1	5.5	-13.1		
$\langle 2,3,4 \rangle$	[1]	24	20	20.0	11.9	-8.0		
$\langle 2,4,3 \rangle$	[10]	24	20	20.0	4.8	-15.3		
$\langle 2,5,2 \rangle$	[10]	20	18	11.1	1.5	-23.1		
$\langle 3,2,2 \rangle$	[10]	12	11	9.1	7.1	-6.6		
$\langle 3,2,3 \rangle$	[10]	18	15	20.0	14.1	-0.7		
$\langle 3,2,4 \rangle$	[10]	24	20	20.0	11.9	-1.8		
$\langle 3,3,2 \rangle$	[10]	18	15	20.0	11.4	-8.1		
$\langle 3,3,3 \rangle$	[14]	27	23	17.4	8.6	-9.3		
$\langle 3,3,6 \rangle$	[14]	54	40	35.0	-34.0	-41.6		
$\langle 3,4,2 \rangle$	[1]	24	20	20.0	4.9	-15.7		
$\langle 3,4,3 \rangle$	[14]	36	29	24.1	8.4	-12.6		
$\langle 3,5,3 \rangle$	[14]	45	36	25.0	5.2	-20.6		
$\langle 3,6,3 \rangle$	[14]	54	40	35.0	-21.6	-64.5		
$\langle 4,2,2 \rangle$	[10]	16	14	14.3	9.4	-4.7		
$\langle 4,2,3 \rangle$	[1]	24	20	20.0	12.1	-2.3		
$\langle 4,2,4 \rangle$	[10]	32	26	23.1	10.4	-2.7		
$\langle 4,3,2 \rangle$	[10]	24	20	20.0	11.3	-7.8		
$\langle 4,3,3 \rangle$	[10]	36	29	24.1	8.1	-8.4		
$\langle 4,4,2 \rangle$	[10]	32	26	23.1	-4.2	-18.4		
$\langle 5,2,2 \rangle$	[10]	20	18	11.1	7.0	-6.7		
$\langle 6,3,3 \rangle$	[14]	54	40	35.0	-33.4	-42.2		



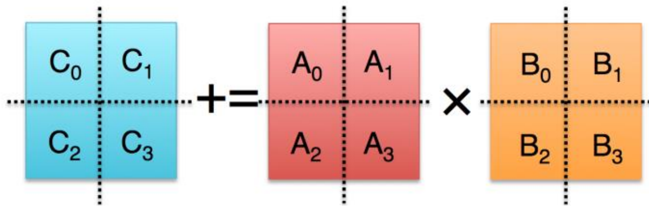
* [1] Austin R. Benson, Grey Ballard. "A framework for practical parallel fast matrix multiplication." *PPoPP*15.

$\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$	Ref.	$\widetilde{m}\widetilde{k}\widetilde{n}$	R	Speedup (%) Theory	Practical #1 Ours	Practical #1 [1]	Practical #2 Ours	Practical #2 [1]
$\langle 2,2,2 \rangle$	[13]	8	7	14.3	11.9	-3.0	13.1	13.1
$\langle 2,3,2 \rangle$	[1]	12	11	9.1	5.5	-13.1	7.7	7.7
$\langle 2,3,4 \rangle$	[1]	24	20	20.0	11.9	-8.0	16.3	17.0
$\langle 2,4,3 \rangle$	[10]	24	20	20.0	4.8	-15.3	14.9	16.6
$\langle 2,5,2 \rangle$	[10]	20	18	11.1	1.5	-23.1	8.6	8.3
$\langle 3,2,2 \rangle$	[10]	12	11	9.1	7.1	-6.6	7.2	7.5
$\langle 3,2,3 \rangle$	[10]	18	15	20.0	14.1	-0.7	17.2	16.8
$\langle 3,2,4 \rangle$	[10]	24	20	20.0	11.9	-1.8	16.1	17.0
$\langle 3,3,2 \rangle$	[10]	18	15	20.0	11.4	-8.1	17.3	16.5
$\langle 3,3,3 \rangle$	[14]	27	23	17.4	8.6	-9.3	14.4	14.7
$\langle 3,3,6 \rangle$	[14]	54	40	35.0	-34.0	-41.6	24.2	20.1
$\langle 3,4,2 \rangle$	[1]	24	20	20.0	4.9	-15.7	16.0	16.8
$\langle 3,4,3 \rangle$	[14]	36	29	24.1	8.4	-12.6	18.1	20.1
$\langle 3,5,3 \rangle$	[14]	45	36	25.0	5.2	-20.6	19.1	18.9
$\langle 3,6,3 \rangle$	[14]	54	40	35.0	-21.6	-64.5	19.5	17.8
$\langle 4,2,2 \rangle$	[10]	16	14	14.3	9.4	-4.7	11.9	12.2
$\langle 4,2,3 \rangle$	[1]	24	20	20.0	12.1	-2.3	15.9	17.3
$\langle 4,2,4 \rangle$	[10]	32	26	23.1	10.4	-2.7	18.4	19.1
$\langle 4,3,2 \rangle$	[10]	24	20	20.0	11.3	-7.8	16.8	15.7
$\langle 4,3,3 \rangle$	[10]	36	29	24.1	8.1	-8.4	19.8	20.0
$\langle 4,4,2 \rangle$	[10]	32	26	23.1	-4.2	-18.4	17.1	18.5
$\langle 5,2,2 \rangle$	[10]	20	18	11.1	7.0	-6.7	8.2	8.5
$\langle 6,3,3 \rangle$	[14]	54	40	35.0	-33.4	-42.2	24.0	20.2



* [1] Austin R. Benson, Grey Ballard. "A framework for practical parallel fast matrix multiplication." *PPoPP*15.

One-level Fast Matrix Multiplication (FMM)



$$C = \left(\begin{array}{c|c|c} C_0 & \cdots & C_{\widetilde{n}-1} \\ \hline \vdots & & \vdots \\ \hline C_{(\widetilde{m}-1)\widetilde{n}} & \cdots & C_{\widetilde{m}\widetilde{n}-1} \end{array}\right), A = \left(\begin{array}{c|c|c} A_0 & \cdots & A_{\widetilde{k}-1} \\ \hline \vdots & & \vdots \\ \hline A_{(\widetilde{m}-1)\widetilde{k}} & \cdots & A_{\widetilde{m}\widetilde{k}-1} \end{array}\right), B = \left(\begin{array}{c|c|c} B_0 & \cdots & B_{\widetilde{n}-1} \\ \hline \vdots & & \vdots \\ \hline B_{(\widetilde{k}-1)\widetilde{n}} & \cdots & B_{\widetilde{k}\widetilde{n}-1} \end{array}\right)$$

for $\quad r = 0, ..., R-1,$

$$M_r := \left(\sum_{i=0}^{\widetilde{m}\widetilde{k}-1} u_{ir} A_i\right) \times \left(\sum_{j=0}^{\widetilde{k}\widetilde{n}-1} v_{jr} B_j\right);$$

$$C_p += w_{pr} M_r \ (p = 0, ..., \widetilde{m}\widetilde{n} - 1)$$

The set of coefficients that determine the $\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$ algorithm is denoted as $[\![U, V, W]\!]$.

Two-level Fast Matrix Multiplication (FMM)

$$C = \left(\begin{array}{c|c|c} C_0 & \cdots & C_{\widetilde{n}-1} \\ \hline \vdots & & \vdots \\ \hline C_{(\widetilde{m}-1)\widetilde{n}} & \cdots & C_{\widetilde{m}\widetilde{n}-1} \end{array}\right), A = \left(\begin{array}{c|c|c} A_0 & \cdots & A_{\widetilde{k}-1} \\ \hline \vdots & & \vdots \\ \hline A_{(\widetilde{m}-1)\widetilde{k}} & \cdots & A_{\widetilde{m}\widetilde{k}-1} \end{array}\right), B = \left(\begin{array}{c|c|c} B_0 & \cdots & B_{\widetilde{n}-1} \\ \hline \vdots & & \vdots \\ \hline B_{(\widetilde{k}-1)\widetilde{n}} & \cdots & B_{\widetilde{k}\widetilde{n}-1} \end{array}\right)$$

The set of coefficients of a two-level $\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$ and $\langle \widetilde{m'}, \widetilde{k'}, \widetilde{n'} \rangle$ FMM algorithm can be denoted as $\boxed{}$.

Kronecker Product

$$X \otimes Y = \begin{pmatrix} x_{0,0}Y & \cdots & x_{0,n-1}Y \\ \vdots & \ddots & \vdots \\ x_{m-1,0}Y & \cdots & x_{m-1,n-1}Y \end{pmatrix}$$

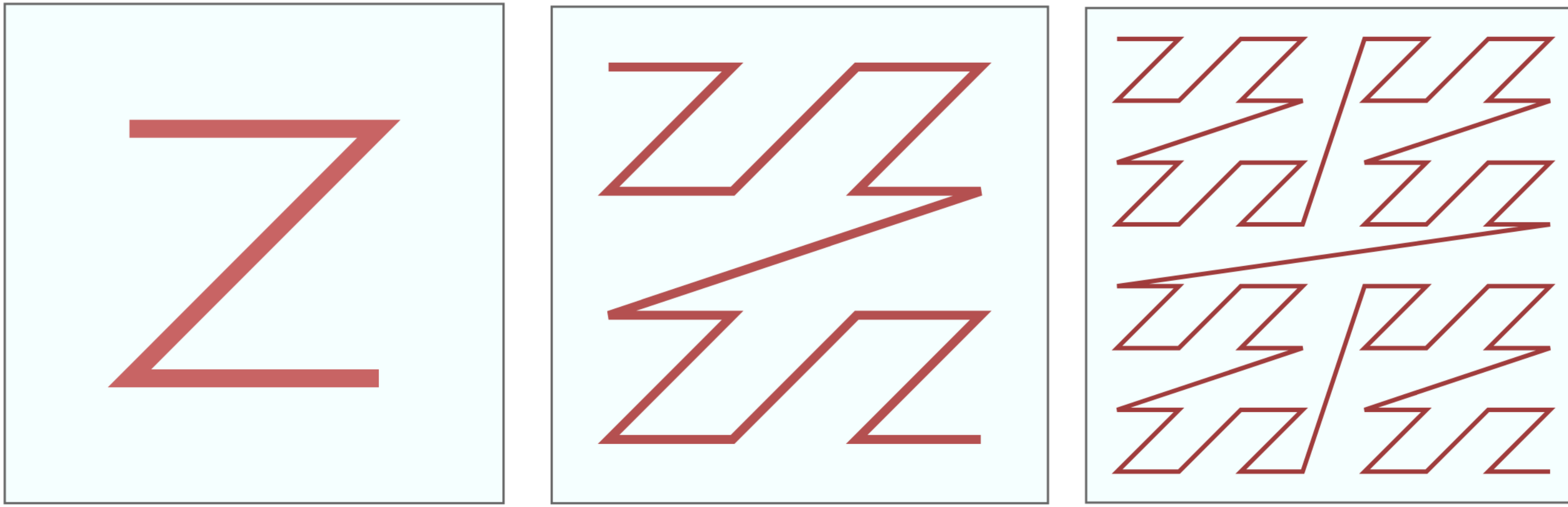
m* x *n ***p* x *q*** ***(mp)* x *(nq)***

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \otimes \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} a_{1,1}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} & a_{1,2}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \\ a_{2,1}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} & a_{2,2}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & a_{2,2}b_{2,1} & a_{2,2}b_{2,2} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1\cdot 0 & 1\cdot 5 & 2\cdot 0 & 2\cdot 5 \\ 1\cdot 6 & 1\cdot 7 & 2\cdot 6 & 2\cdot 7 \\ 3\cdot 0 & 3\cdot 5 & 4\cdot 0 & 4\cdot 5 \\ 3\cdot 6 & 3\cdot 7 & 4\cdot 6 & 4\cdot 7 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}$$

- https://en.wikipedia.org/wiki/Kronecker_product.
- https://en.wikipedia.org/wiki/Tensor_product

Morton-like Ordering



$$
\begin{pmatrix}
0 & 1 & 4 & 5 & 16 & 17 & 20 & 21 \\
2 & 3 & 6 & 7 & 18 & 19 & 22 & 23 \\
8 & 9 & 12 & 13 & 24 & 25 & 28 & 29 \\
10 & 11 & 14 & 15 & 26 & 27 & 30 & 31 \\
32 & 33 & 36 & 37 & 48 & 49 & 52 & 53 \\
34 & 35 & 38 & 39 & 50 & 51 & 54 & 55 \\
40 & 41 & 44 & 45 & 56 & 57 & 60 & 61 \\
42 & 43 & 46 & 47 & 58 & 59 & 62 & 63
\end{pmatrix}
$$

* https://en.wikipedia.org/wiki/Z-order_curve.

Two-level Fast Matrix Multiplication

$$C = \left(\begin{array}{c|c|c} C_0 & \cdots & C_{\widetilde{n}-1} \\ \hline \vdots & & \vdots \\ \hline C_{(\widetilde{m}-1)\widetilde{n}} & \cdots & C_{\widetilde{m}\widetilde{n}-1} \end{array} \right), A = \left(\begin{array}{c|c|c} A_0 & \cdots & A_{\widetilde{k}-1} \\ \hline \vdots & & \vdots \\ \hline A_{(\widetilde{m}-1)\widetilde{k}} & \cdots & A_{\widetilde{m}\widetilde{k}-1} \end{array} \right), B = \left(\begin{array}{c|c|c} B_0 & \cdots & B_{\widetilde{n}-1} \\ \hline \vdots & & \vdots \\ \hline B_{(\widetilde{k}-1)\widetilde{n}} & \cdots & B_{\widetilde{k}\widetilde{n}-1} \end{array} \right)$$

The set of coefficients of a two-level $\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$ and $\langle \widetilde{m'}, \widetilde{k'}, \widetilde{n'} \rangle$ FMM algorithm can be denoted as $[\![U \,?\, U', V \,?\, V', W \,?\, W']\!]$.

Two-level Fast Matrix Multiplication

for $r = 0, \ldots, R \cdot R' - 1,$

$$M_r := \left(\sum_{i=0}^{\widetilde{m}\widetilde{k}\cdot\widetilde{m}'\widetilde{k}'-1} \boxed{(U \otimes U')}_{i,r} A_i \right) \times \left(\sum_{j=0}^{\widetilde{k}\widetilde{n}\cdot\widetilde{k}'\widetilde{n}'-1} \boxed{(V \otimes V')}_{j,r} B_j \right);$$

$$C_p {+}{=} \boxed{(W \otimes W')}_{p,r} M_r (p = 0, \ldots, \widetilde{m}\widetilde{n} \cdot \widetilde{m}'\widetilde{n}' - 1)$$

2-level Morton-like ordering index

The set of coefficients of a two-level $\langle \widetilde{m}, \widetilde{k}, \widetilde{n} \rangle$ and $\langle \widetilde{m'}, \widetilde{k'}, \widetilde{n'} \rangle$ FMM algorithm can be denoted as $[\![U \otimes U', V \otimes V', W \otimes W']\!]$.

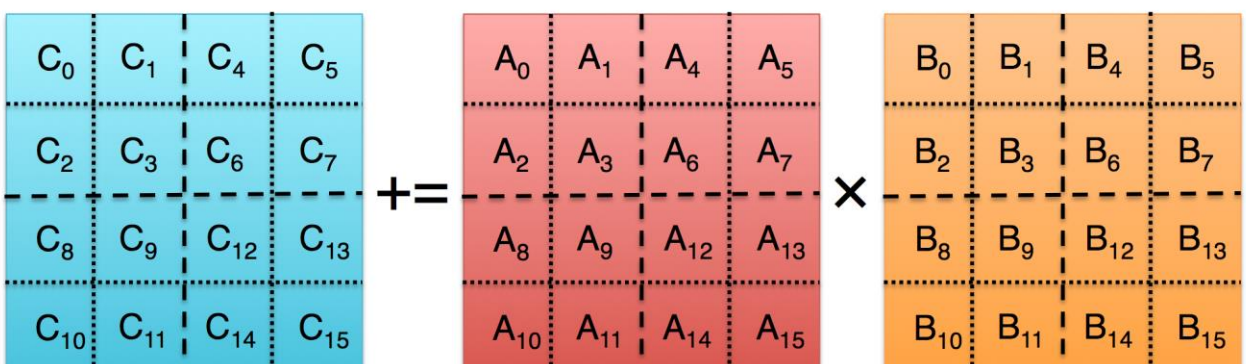
Two-level <2,2,2> Strassen Algorithm

$M_0 := (A_0+A_{12}+A_3+A_{15})(B_0+B_{12}+B_3+B_{15});$	$C_0 \mathrel{+}= M_0;$	$C_3 \mathrel{+}= M_0;$	$C_{12} \mathrel{+}= M_0;$	$C_{15} \mathrel{+}= M_0;$
$M_1 := (A_2+A_{14}+A_3+A_{15})(B_0+B_{12});$	$C_2 \mathrel{+}= M_1;$	$C_3 \mathrel{-}= M_1;$	$C_{14} \mathrel{+}= M_1;$	$C_{15} \mathrel{-}= M_1;$
$M_2 := (A_0+A_{12})(B_1+B_{13}+B_3+B_{15});$	$C_1 \mathrel{+}= M_2;$	$C_3 \mathrel{+}= M_2;$	$C_{13} \mathrel{+}= M_2;$	$C_{15} \mathrel{+}= M_2;$
$M_3 := (A_3+A_{15})(B_2+B_{14}+B_0+B_{12});$	$C_0 \mathrel{+}= M_3;$	$C_2 \mathrel{+}= M_3;$	$C_{12} \mathrel{+}= M_3;$	$C_{14} \mathrel{+}= M_3;$
$M_4 := (A_0+A_{12}+A_1+A_{13})(B_3+B_{15});$	$C_0 \mathrel{-}= M_4;$	$C_1 \mathrel{+}= M_4;$	$C_{12} \mathrel{-}= M_4;$	$C_{13} \mathrel{+}= M_4;$
$M_5 := (A_2+A_{14}+A_0+A_{12})(B_0+B_{12}+B_1+B_{13});$	$C_3 \mathrel{+}= M_5;$	$C_{15} \mathrel{+}= M_5;$		
$M_6 := (A_1+A_{13}+A_3+A_{15})(B_2+B_{14}+B_3+B_{15});$	$C_0 \mathrel{+}= M_6;$	$C_{12} \mathrel{+}= M_6;$		
$M_7 := (A_8+A_{12}+A_{11}+A_{15})(B_0+B_3);$	$C_8 \mathrel{+}= M_7;$	$C_{11} \mathrel{+}= M_7;$	$C_{12} \mathrel{-}= M_7;$	$C_{15} \mathrel{-}= M_7;$
$M_8 := (A_{10}+A_{14}+A_{11}+A_{15})(B_0);$	$C_{10} \mathrel{+}= M_8;$	$C_{11} \mathrel{-}= M_8;$	$C_{14} \mathrel{-}= M_8;$	$C_{15} \mathrel{+}= M_8;$
$M_9 := (A_8+A_{12})(B_1+B_3);$	$C_9 \mathrel{+}= M_9;$	$C_{11} \mathrel{+}= M_9;$	$C_{13} \mathrel{-}= M_9;$	$C_{15} \mathrel{-}= M_9;$
$M_{10} := (A_{11}+A_{15})(B_2+B_0);$	$C_8 \mathrel{+}= M_{10};$	$C_{10} \mathrel{+}= M_{10};$	$C_{12} \mathrel{-}= M_{10};$	$C_{14} \mathrel{-}= M_{10};$
$\vdots$				
$M_{40} := (A_{10}+A_2+A_8+A_0)(B_0+B_4+B_1+B_5);$	$C_{15} \mathrel{+}= M_{40};$			
$M_{41} := (A_9+A_1+A_{11}+A_3)(B_2+B_6+B_3+B_7);$	$C_{12} \mathrel{+}= M_{41};$			
$M_{42} := (A_4+A_{12}+A_7+A_{15})(B_8+B_{12}+B_{11}+B_{15});$	$C_0 \mathrel{+}= M_{42};$	$C_3 \mathrel{+}= M_{42};$		
$M_{43} := (A_6+A_{14}+A_7+A_{15})(B_8+B_{12});$	$C_2 \mathrel{+}= M_{43};$	$C_3 \mathrel{-}= M_{43};$		
$M_{44} := (A_4+A_{12})(B_9+B_{13}+B_{11}+B_{15});$	$C_1 \mathrel{+}= M_{44};$	$C_3 \mathrel{+}= M_{44};$		
$M_{45} := (A_7+A_{15})(B_{10}+B_{14}+B_8+B_{12});$	$C_0 \mathrel{+}= M_{45};$	$C_2 \mathrel{+}= M_{45};$		
$M_{46} := (A_4+A_{12}+A_5+A_{13})(B_{11}+B_{15});$	$C_0 \mathrel{-}= M_{46};$	$C_1 \mathrel{+}= M_{46};$		
$M_{47} := (A_6+A_{14}+A_4+A_{12})(B_8+B_{12}+B_9+B_{13});$	$C_3 \mathrel{+}= M_{47};$			
$M_{48} := (A_5+A_{13}+A_7+A_{15})(B_{10}+B_{14}+B_{11}+B_{15});$	$C_0 \mathrel{+}= M_{48};$			

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$



$$\boxed{[\![U \otimes U, V \otimes V, W \otimes W]\!]}$$

L-level Fast Matrix Multiplication

for $\quad r = 0, \ldots, \prod_{l=0}^{L-1} R_l - 1,$

$$M_r := \left(\sum_{i=0}^{\prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{k}_l - 1} \boxed{\left(\bigotimes_{l=0}^{L-1} U_l\right)}_{i,r} A_i \right) \times \left(\sum_{j=0}^{\prod_{l=0}^{L-1} \widetilde{k}_l \widetilde{n}_l - 1} \boxed{\left(\bigotimes_{l=0}^{L-1} V_l\right)}_{j,r} B_j \right);$$

$$C_p += \boxed{\left(\bigotimes_{l=0}^{L-1} W_l\right)}_{p,r} M_r \left(p = 0, \ldots, \prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{n}_l - 1\right)$$

***L*-level Morton-like ordering index**

The set of coefficients of an L-level $\langle \widetilde{m}_l, \widetilde{k}_l, \widetilde{n}_l \rangle$ $(l{=}0, 1, \ldots, L{-}1)$ FMM algorithm can be denoted as $[\![\bigotimes_{l=0}^{L-1} U_l, \bigotimes_{l=0}^{L-1} V_l, \bigotimes_{l=0}^{L-1} W_l]\!]$.

Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
- Fast Matrix Multiplication (FMM)
- Code Generation
- Performance Model
- Experiments
- Conclusion

Generating skeleton framework

- Compute the Kronecker Product.

$$[\![\bigotimes_{l=0}^{L-1} U_l, \bigotimes_{l=0}^{L-1} V_l, \bigotimes_{l=0}^{L-1} W_l]\!]$$

- Generate matrix partitions by conceptually morton-like ordering indexing.

$$\langle \widetilde{m_l}, \widetilde{k_l}, \widetilde{n_l} \rangle$$

- Fringes: dynamic peeling.

$$\prod_{l=0}^{L-1} \widetilde{m_l}, \prod_{l=0}^{L-1} \widetilde{k_l}, \prod_{l=0}^{L-1} \widetilde{n_l}$$

$$A = \left(\begin{array}{c|c} A_{11} & a_{12} \\ \hline a_{21} & a_{22} \end{array}\right) \quad B = \left(\begin{array}{c|c} B_{11} & b_{12} \\ \hline b_{21} & b_{22} \end{array}\right) \quad C = \left(\begin{array}{c|c} C_{11} & c_{12} \\ \hline c_{21} & c_{22} \end{array}\right)$$

$$C_{11} = a_{12} b_{21} + C_{11} \qquad c_{12} = \begin{pmatrix} A_{11} & a_{12} \end{pmatrix} \begin{pmatrix} b_{12} \\ b_{22} \end{pmatrix} \qquad \begin{pmatrix} c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{21} & a_{22} \end{pmatrix} B$$

*Mithuna Thottethodi, Siddhartha Chatterjee, and Alvin R. Lebeck. "Tuning Strassen's matrix multiplication for memory efficiency." *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 1998.

Generating typical operations

$$M_r := \left(\prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{k}_l - 1 \sum_{i=0} \left(\bigotimes_{l=0}^{L-1} U_l\right)_{i,r} A_i \right) \times \left(\prod_{l=0}^{L-1} \widetilde{k}_l \widetilde{n}_l - 1 \sum_{j=0} \left(\bigotimes_{l=0}^{L-1} V_l\right)_{j,r} B_j \right);$$

$$C_p += \left(\bigotimes_{l=0}^{L-1} W_l\right)_{p,r} M_r \left(p = 0, \ldots, \prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{n}_l - 1\right)$$

- Packing routines:

$$\prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{k}_l - 1 \sum_{i=0} \left(\bigotimes_{l=0}^{L-1} U_l\right)_{i,r} A_i \rightarrow \widetilde{A}_i$$

$$\prod_{l=0}^{L-1} \widetilde{k}_l \widetilde{n}_l - 1 \sum_{j=0} \left(\bigotimes_{l=0}^{L-1} V_l\right)_{j,r} B_j \rightarrow \widetilde{B}_p$$

- Micro-kernel:
  - A hand-coded GEMM kernel
  - Automatically generated updates to multiple submatrices of C.

$$C_p += \left(\bigotimes_{l=0}^{L-1} W_l\right)_{p,r} M_r \left(p = 0, \ldots, \prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{n}_l - 1\right)$$

Variations on a theme

- Naïve FMM

$$M_r := \left(\sum_{i=0}^{\prod_{l=0}^{L-1} \widetilde{m_l}\widetilde{k_l}-1} (\bigotimes_{l=0}^{L-1} U_l)_{i,r} A_i \right) \times \left(\sum_{j=0}^{\prod_{l=0}^{L-1} \widetilde{k_l}\widetilde{n_l}-1} (\bigotimes_{l=0}^{L-1} V_l)_{j,r} B_j \right);$$

$$C_p += (\bigotimes_{l=0}^{L-1} W_l)_{p,r} M_r (p = 0, \ldots, \prod_{l=0}^{L-1} \widetilde{m_l}\widetilde{n_l} - 1)$$

  – A traditional implementation with temporary buffers.

- AB FMM
  – Integrate the addition of matrices into $\widetilde{A_i}$ and $\widetilde{B_p}$.

$$\sum_{i=0}^{\prod_{l=0}^{L-1} \widetilde{m_l}\widetilde{k_l}-1} (\bigotimes_{l=0}^{L-1} U_l)_{i,r} A_i \rightarrow \widetilde{A_i}$$

$$\sum_{j=0}^{\prod_{l=0}^{L-1} \widetilde{k_l}\widetilde{n_l}-1} (\bigotimes_{l=0}^{L-1} V_l)_{j,r} B_j \rightarrow \widetilde{B_p}$$

- ABC FMM
  – 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. $C_p += (\bigotimes_{l=0}^{L-1} W_l)_{p,r} M_r (p = 0, \ldots, \prod_{l=0}^{L-1} \widetilde{m_l}\widetilde{n_l} - 1)$

Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
- Fast Matrix Multiplication (FMM)
- Code Generation
- Performance Model
- Experiments
- Conclusion

Performance Model

- Performance Metric

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

- Total Time Breakdown

$$T = T_a + T_m$$

Arithmetic Operations

Memory Operations

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

	type	τ	GEMM	L-level
$T_a^{\times}$	-	τ_a	$2mnk$	$2\frac{m}{\widetilde{M_L}}\frac{n}{\widetilde{N_L}}\frac{k}{\widetilde{K_L}}$
T_a^{A+}	-	τ_a	-	$2\frac{m}{\widetilde{M_L}}\frac{k}{\widetilde{K_L}}$
T_a^{B+}	-	τ_a	-	$2\frac{k}{\widetilde{K_L}}\frac{n}{\widetilde{N_L}}$
T_a^{C+}	-	τ_a	-	$2\frac{m}{\widetilde{M_L}}\frac{n}{\widetilde{N_L}}$

	GEMM	L-level		
		ABC	AB	Naive
$N_a^{\times}$	1	R_L	R_L	R_L
N_a^{A+}	-	$nnz(\bigotimes U)$-R_L	$nnz(\bigotimes U)$-R_L	$nnz(\bigotimes U)$-R_L
N_a^{B+}	-	$nnz(\bigotimes V)$-R_L	$nnz(\bigotimes V)$-R_L	$nnz(\bigotimes V)$-R_L
N_a^{C+}	-	$nnz(\bigotimes W)$	$nnz(\bigotimes W)$	$nnz(\bigotimes W)$

$\widetilde{M_L} = \prod_{l=0}^{L-1} \widetilde{m}_l$, $\widetilde{K_L} = \prod_{l=0}^{L-1} \widetilde{k}_l$, $\widetilde{N_L} = \prod_{l=0}^{L-1} \widetilde{n}_l$, $R_L = \prod_{l=0}^{L-1} R_l$.

$\bigotimes U = \bigotimes_{l=0}^{L-1} U_l$, $\bigotimes V = \bigotimes_{l=0}^{L-1} V_l$, $\bigotimes W = \bigotimes_{l=0}^{L-1} W_l$.

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

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

	type	τ	GEMM	L-level
$T_a^{\times}$	-	τ_a	$2mnk$	$2\frac{m}{\widetilde{M_L}}\frac{n}{\widetilde{N_L}}\frac{k}{\widetilde{K_L}}$
T_a^{A+}	-	τ_a	-	$2\frac{m}{\widetilde{M_L}}\frac{k}{\widetilde{K_L}}$
T_a^{B+}	-	τ_a	-	$2\frac{k}{\widetilde{K_L}}\frac{n}{\widetilde{N_L}}$
T_a^{C+}	-	τ_a	-	$2\frac{m}{\widetilde{M_L}}\frac{n}{\widetilde{N_L}}$
$T_m^{A\times}$	r	τ_b	$mk\lceil\frac{n}{n_c}\rceil$	$\frac{m}{\widetilde{M_L}}\frac{k}{\widetilde{K_L}}\lceil\frac{n/\widetilde{N_L}}{n_c}\rceil$
$\widetilde{T}_m^{A\times}$	w	τ_b	$mk\lceil\frac{n}{n_c}\rceil$	$\frac{m}{\widetilde{M_L}}\frac{k}{\widetilde{K_L}}\lceil\frac{n/\widetilde{N_L}}{n_c}\rceil$
$T_m^{B\times}$	r	τ_b	nk	$\frac{n}{\widetilde{N_L}}\frac{k}{\widetilde{K_L}}$
$\widetilde{T}_m^{B\times}$	w	τ_b	nk	$\frac{n}{\widetilde{N_L}}\frac{k}{\widetilde{K_L}}$
$T_m^{C\times}$	r/w	τ_b	$2\lambda mn\lceil\frac{k}{k_c}\rceil$	$2\lambda\frac{m}{\widetilde{M_L}}\frac{n}{\widetilde{N_L}}\lceil\frac{k/\widetilde{K_L}}{k_c}\rceil$
T_m^{A+}	r/w	τ_b	mk	$\frac{m}{\widetilde{M_L}}\frac{k}{\widetilde{K_L}}$
T_m^{B+}	r/w	τ_b	nk	$\frac{n}{\widetilde{N_L}}\frac{k}{\widetilde{K_L}}$
T_m^{C+}	r/w	τ_b	mn	$\frac{m}{\widetilde{M_L}}\frac{n}{\widetilde{N_L}}$

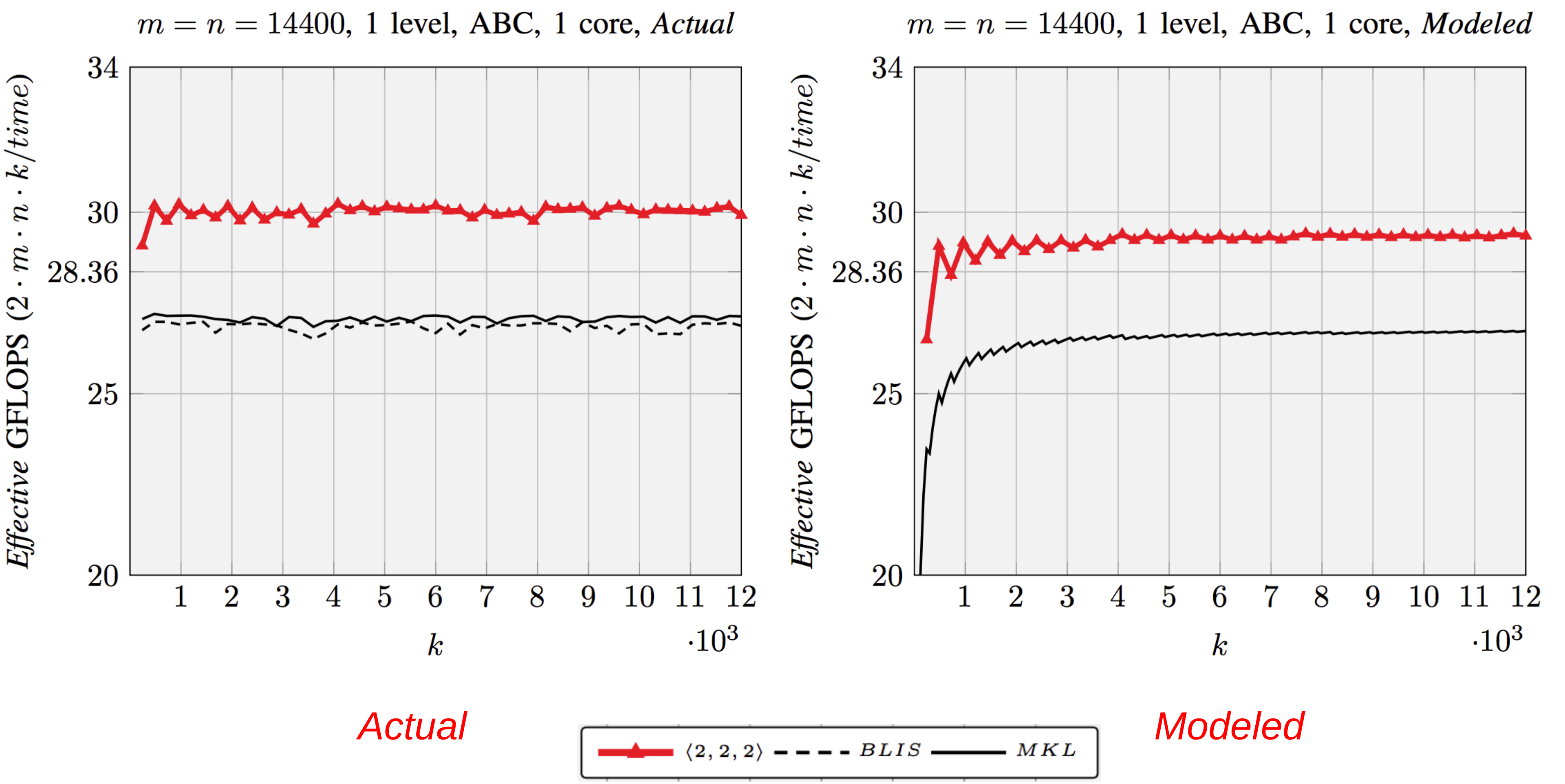
	GEMM	L-level		
		ABC	AB	Naive
$N_a^{\times}$	1	R_L	R_L	R_L
N_a^{A+}	-	$nnz(\bigotimes U)$-R_L	$nnz(\bigotimes U)$-R_L	$nnz(\bigotimes U)$-R_L
N_a^{B+}	-	$nnz(\bigotimes V)$-R_L	$nnz(\bigotimes V)$-R_L	$nnz(\bigotimes V)$-R_L
N_a^{C+}	-	$nnz(\bigotimes W)$	$nnz(\bigotimes W)$	$nnz(\bigotimes W)$
$N_m^{A\times}$	1	$nnz(\bigotimes U)$	$nnz(\bigotimes U)$	R_L
$\widetilde{N}_m^{A\times}$	-	-	-	-
$N_m^{B\times}$	1	$nnz(\bigotimes V)$	$nnz(\bigotimes V)$	R_L
$\widetilde{N}_m^{B\times}$	-	-	-	-
$N_m^{C\times}$	1	$nnz(\bigotimes W)$	R_L	R_L
N_m^{A+}	-	-	-	$nnz(\bigotimes U)+R_L$
N_m^{B+}	-	-	-	$nnz(\bigotimes V)+R_L$
N_m^{C+}	-	-	$3nnz(\bigotimes W)$	$3nnz(\bigotimes W)$

$\widetilde{M_L} = \prod_{l=0}^{L-1}\widetilde{m}_l$, $\widetilde{K_L} = \prod_{l=0}^{L-1}\widetilde{k}_l$, $\widetilde{N_L} = \prod_{l=0}^{L-1}\widetilde{n}_l$, $R_L = \prod_{l=0}^{L-1}R_l$.

$\bigotimes U = \bigotimes_{l=0}^{L-1}U_l$, $\bigotimes V = \bigotimes_{l=0}^{L-1}V_l$, $\bigotimes W = \bigotimes_{l=0}^{L-1}W_l$.

Actual vs. Modeled Performance

Intel Xeon E5-2680 v2 (Ivy Bridge, 10 core/socket)

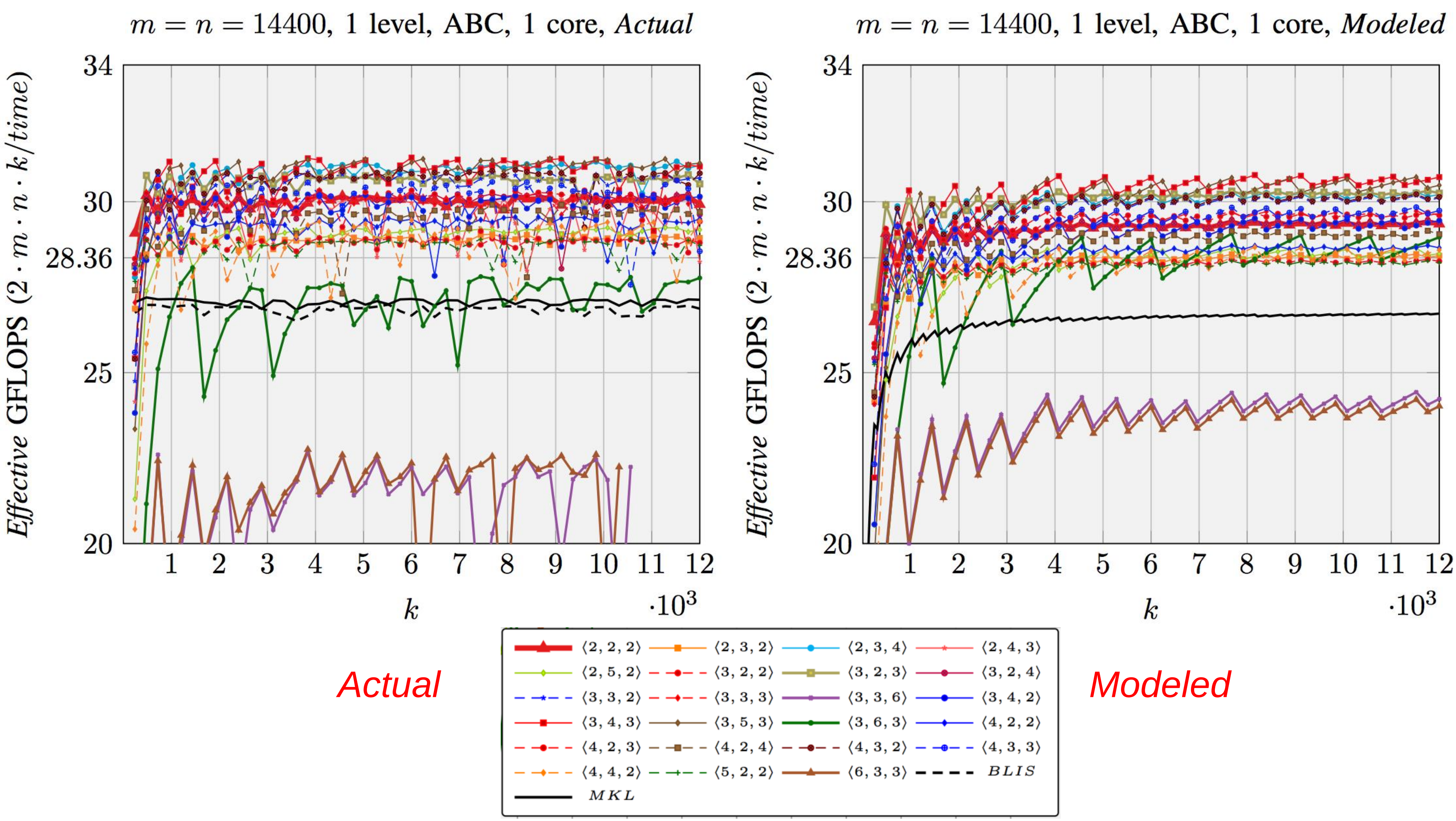


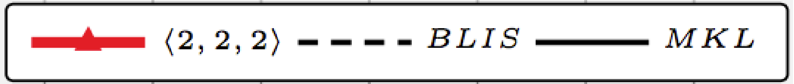
Actual

Modeled

Actual vs. Modeled Performance

Intel Xeon E5-2680 v2 (Ivy Bridge, 10 core/socket)



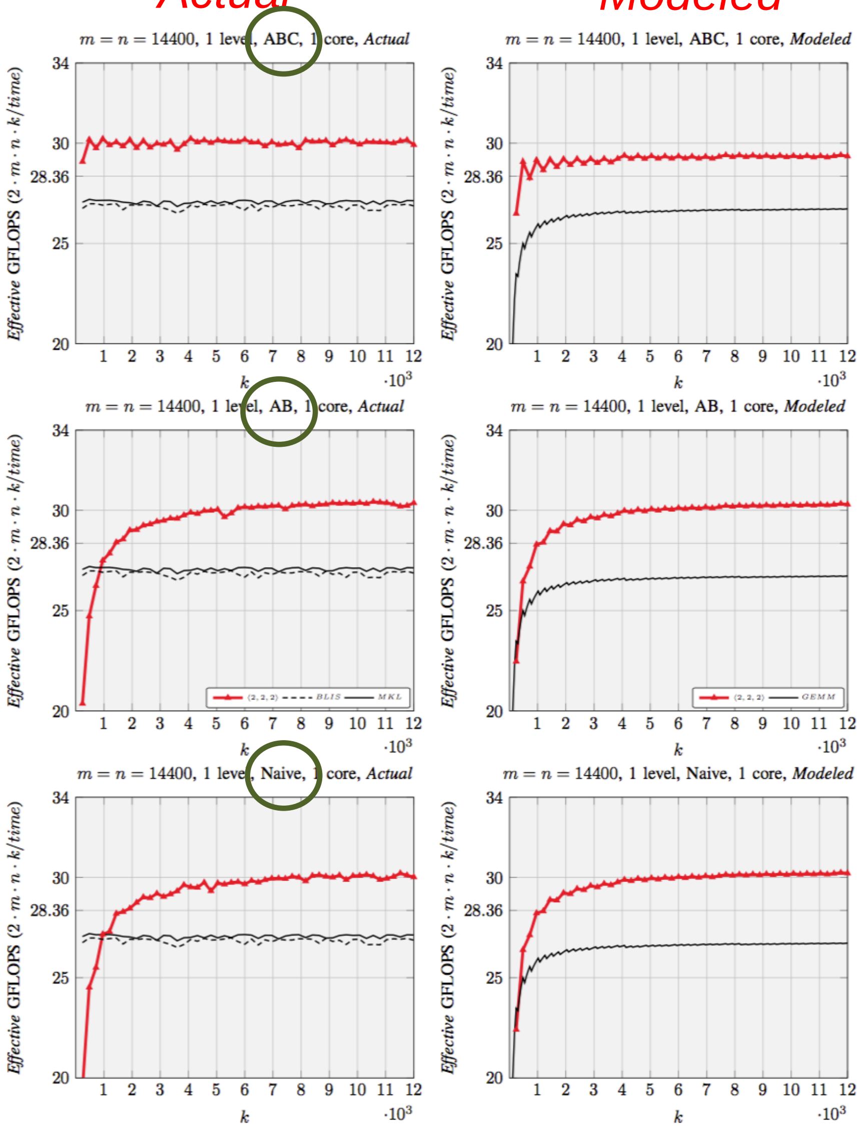


$m = n = 14400$, 1 level

2 level, ABC

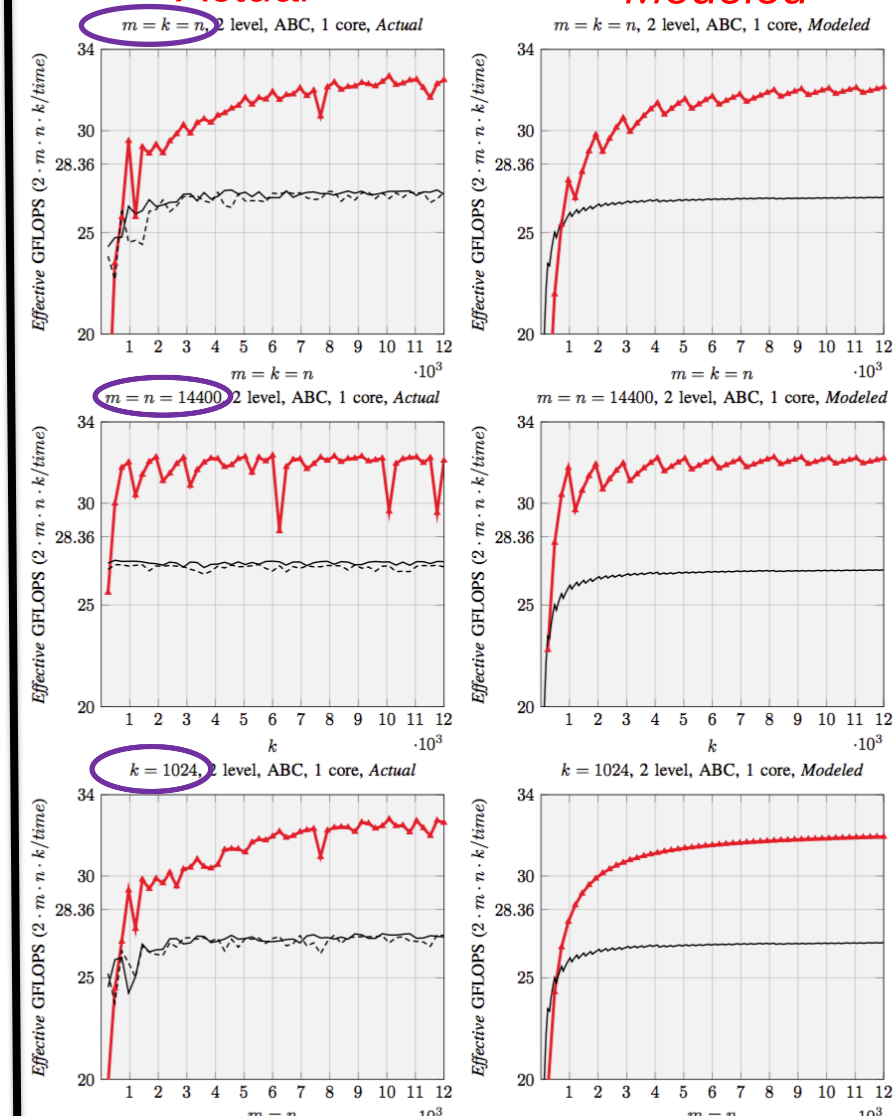
Actual

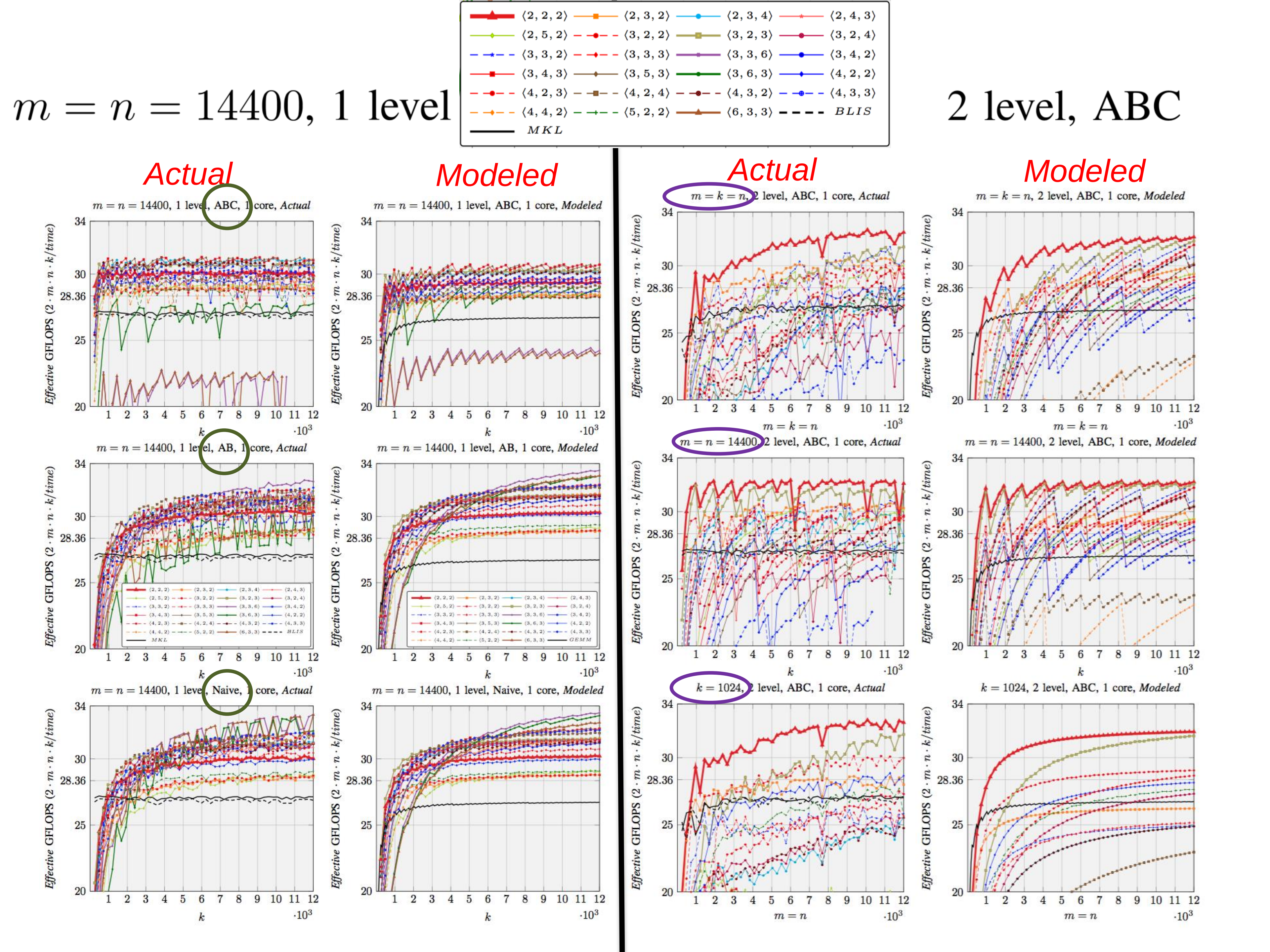
Modeled



Actual

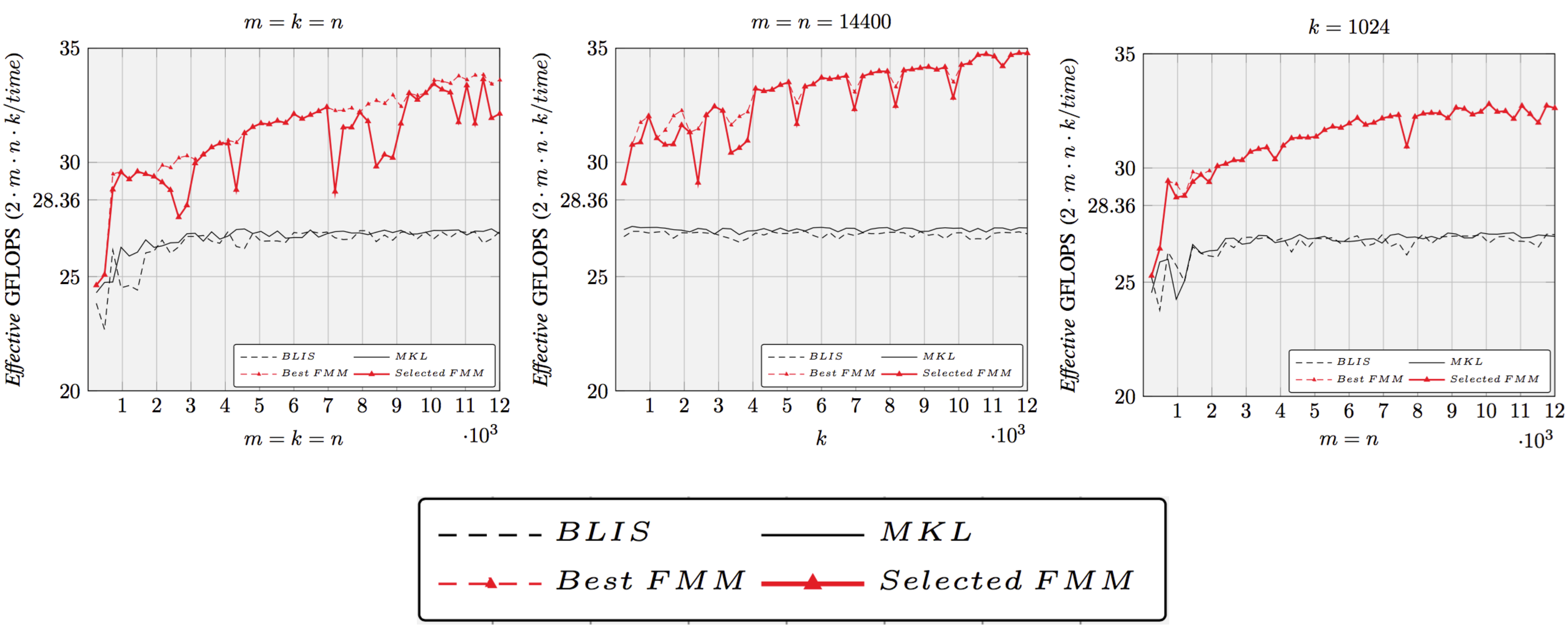
Modeled




m = n = 14400, 1 level
2 level, ABC
⟨2, 2, 2⟩ ⟨2, 3, 2⟩ ⟨2, 3, 4⟩ ⟨2, 4, 3⟩
⟨2, 5, 2⟩ ⟨3, 2, 2⟩ ⟨3, 2, 3⟩ ⟨3, 2, 4⟩
⟨3, 3, 2⟩ ⟨3, 3, 3⟩ ⟨3, 3, 6⟩ ⟨3, 4, 2⟩
⟨3, 4, 3⟩ ⟨3, 5, 3⟩ ⟨3, 6, 3⟩ ⟨4, 2, 2⟩
⟨4, 2, 3⟩ ⟨4, 2, 4⟩ ⟨4, 3, 2⟩ ⟨4, 3, 3⟩
⟨4, 4, 2⟩ ⟨5, 2, 2⟩ ⟨6, 3, 3⟩ BLIS
MKL
Actual
Modeled
Actual
Modeled
m = n = 14400, 1 level, ABC, 1 core, Actual
m = n = 14400, 1 level, ABC, 1 core, Modeled
m = n = 14400, 1 level, AB, 1 core, Actual
m = n = 14400, 1 level, AB, 1 core, Modeled
m = n = 14400, 1 level, Naive, 1 core, Actual
m = n = 14400, 1 level, Naive, 1 core, Modeled
m = k = n, 2 level, ABC, 1 core, Actual
m = k = n, 2 level, ABC, 1 core, Modeled
m = n = 14400, 2 level, ABC, 1 core, Actual
m = n = 14400, 2 level, ABC, 1 core, Modeled
k = 1024, 2 level, ABC, 1 core, Actual
k = 1024, 2 level, ABC, 1 core, Modeled
Effective GFLOPS (2 · m · n · k/time)
k
m = k = n
m = n
·10³

Selecting FMM implementation with performance model

Intel Xeon E5-2680 v2 (Ivy Bridge, 10 core/socket)



Best FMM: Best implementation among 20+ FMM with ABC, AB, Naïve implementations

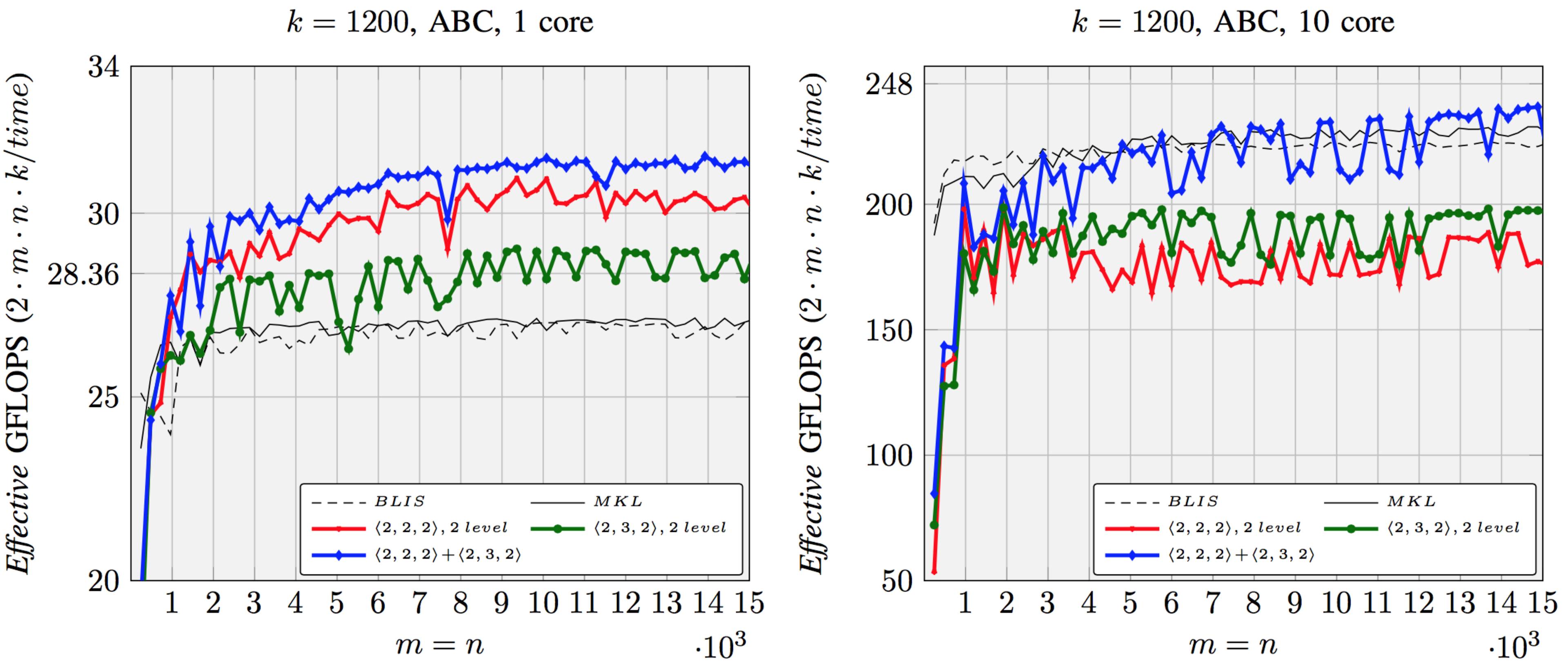
Selected FMM: Selected FMM implementation with the guidance of performance model

Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
- Fast Matrix Multiplication (FMM)
- Code Generation
- Performance Model
- Experiments
- Conclusion

Hybrid Partitions

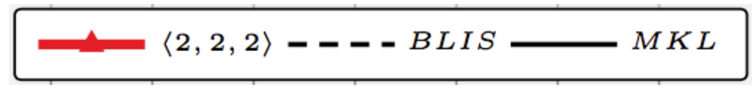
Intel Xeon E5-2680 v2 (Ivy Bridge, 10 core/socket)

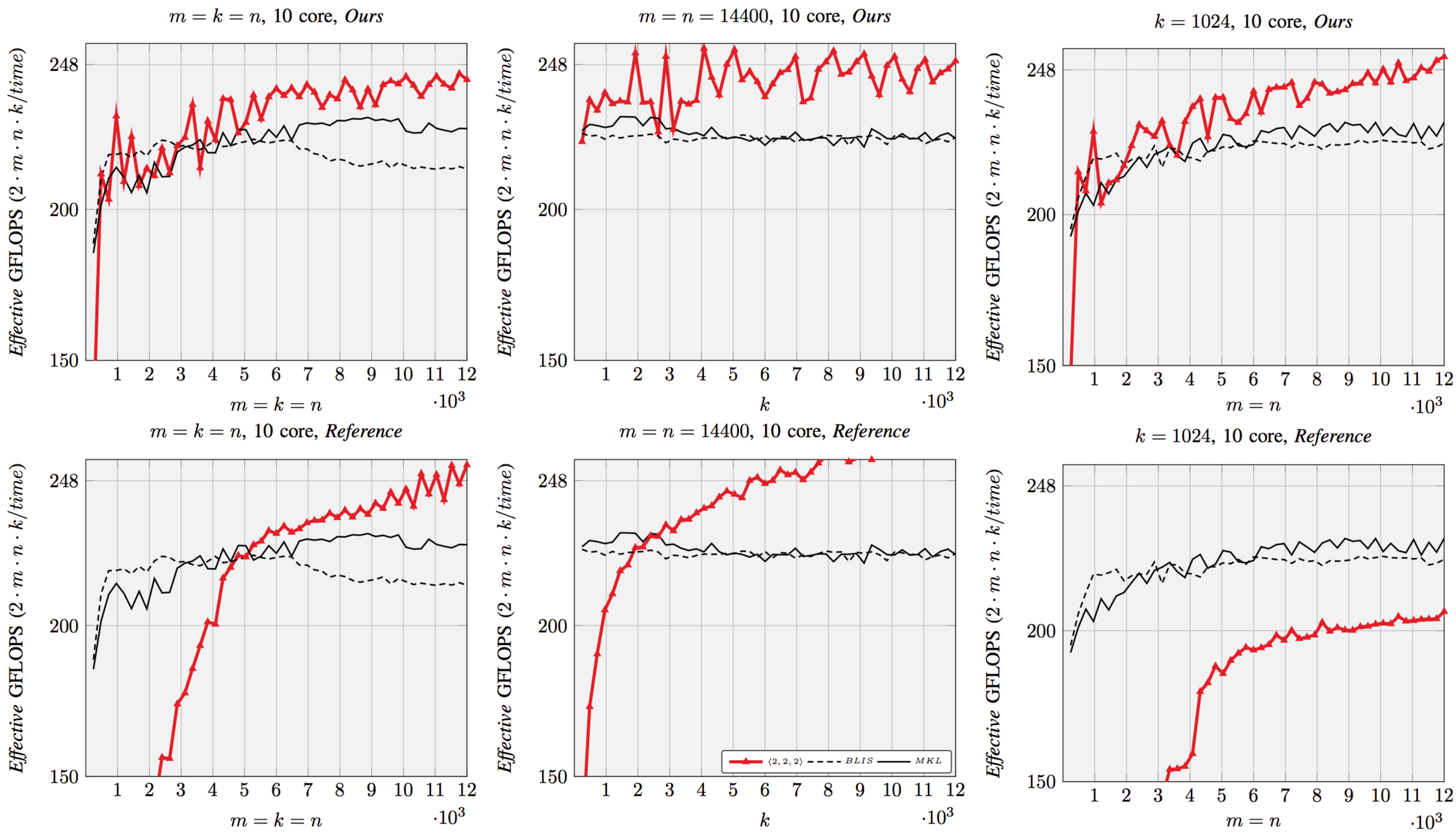


$$[\![U \otimes U', V \otimes V', W \otimes W']\!]$$

Parallel performance

Intel Xeon E5-2680 v2 (Ivy Bridge, 10 core/socket)

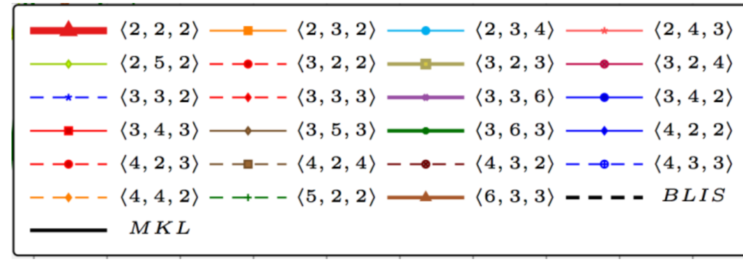


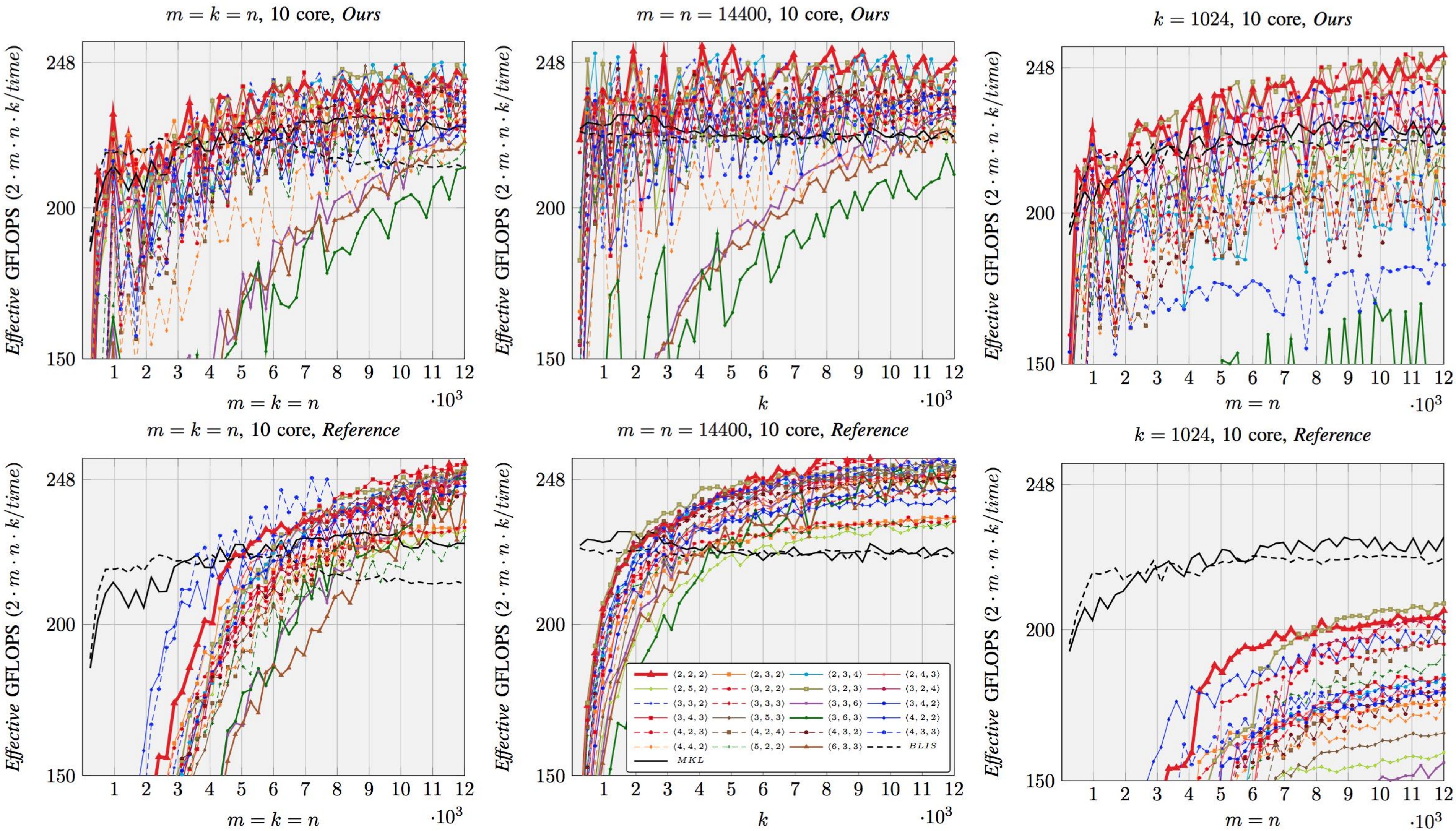


* *Reference*: Austin R. Benson, and Grey Ballard. "A framework for practical parallel fast matrix multiplication." in *PPoPP2015*.

Parallel performance

Intel Xeon E5-2680 v2 (Ivy Bridge, 10 core/socket)





* *Reference*: Austin R. Benson, and Grey Ballard. "A framework for practical parallel fast matrix multiplication." in *PPoPP2015*.

Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
- Fast Matrix Multiplication (FMM)
- Code Generation
- Performance Model
- Experiments
- Conclusion

Our code generator …

- Implement families of FMM algorithms automatically.

$$\langle 2,2,2\rangle \; \langle 2,3,2\rangle \; \langle 2,3,4\rangle \; \langle 2,4,3\rangle \; \langle 2,5,2\rangle \; \langle 3,2,2\rangle \; \ldots \; \langle 6,3,3\rangle$$

- Express multi-level FMM as Kronecker product.

$$[\![U \otimes U', V \otimes V', W \otimes W']\!]$$

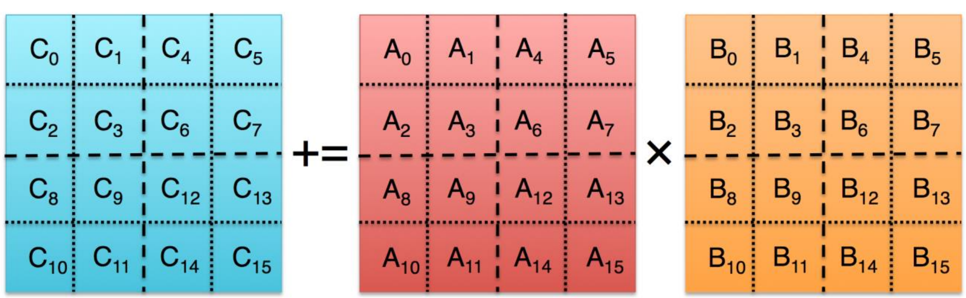
- Incorporate matrix summations into operations inside GEMM.

$$\boxed{\prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{k}_l - 1 \sum_{i=0} (\bigotimes_{l=0}^{L-1} U_l)_{i,r} A_i \rightarrow \widetilde{A}_i}$$

$$\boxed{\prod_{l=0}^{L-1} \widetilde{k}_l \widetilde{n}_l - 1 \sum_{j=0} (\bigotimes_{l=0}^{L-1} V_l)_{j,r} B_j \rightarrow \widetilde{B}_p}$$

$$\boxed{C_p += (\bigotimes_{l=0}^{L-1} W_l)_{p,r} M_r (p = 0, \ldots, \prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{n}_l - 1)}$$

- Generate relatively accurate performance model.



for $r = 0, \ldots, \prod_{l=0}^{L-1} R_l - 1,$

$$M_r := \left(\prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{k}_l - 1 \sum_{i=0} (\bigotimes_{l=0}^{L-1} U_l)_{i,r} A_i \right) \times \left(\prod_{l=0}^{L-1} \widetilde{k}_l \widetilde{n}_l - 1 \sum_{j=0} (\bigotimes_{l=0}^{L-1} V_l)_{j,r} B_j \right);$$

$$C_p += (\bigotimes_{l=0}^{L-1} W_l)_{p,r} M_r (p = 0, \ldots, \prod_{l=0}^{L-1} \widetilde{m}_l \widetilde{n}_l - 1)$$

Acknowledgement

- NSF grants ACI-1550493.
- A gift from Qualcomm Foundation.
- Intel Corporation through an Intel Parallel Computing Center (IPCC).
- Access to the Maverick and Stampede supercomputers administered by TACC.

We thank Austin Benson and Grey Ballard for sharing their open source code and helpful discussions, the anonymous reviewers for their constructive comments, 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!

The source code can be downloaded from:
https://github.com/flame/fmm-gen

Related work

- [1]: Systematic way to identity and implement new FMM algorithms based on conventional calls to GEMM.
- [2]: Strassen can be incorporated into high-performance GEMM
- [3]: Kronecker product for expressing multi-level Strassen's algorithm

*[1] Austin R. Benson, Grey Ballard. "A framework for practical parallel fast matrix multiplication." *PPoPP*15.
*[2] Jianyu Huang, Tyler Smith, Greg Henry, and Robert van de Geijn. "Strassen's Algorithm Reloaded." In *SC'16*.
*[3] C-H Huang, Jeremy R. Johnson, and Robert W. Johnson. "A tensor product formulation of Strassen's matrix multiplication algorithm." Applied Mathematics Letters 3.3 (1990): 67-71.