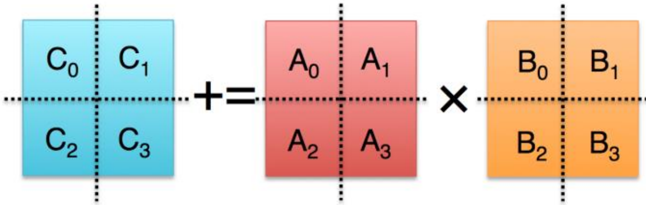
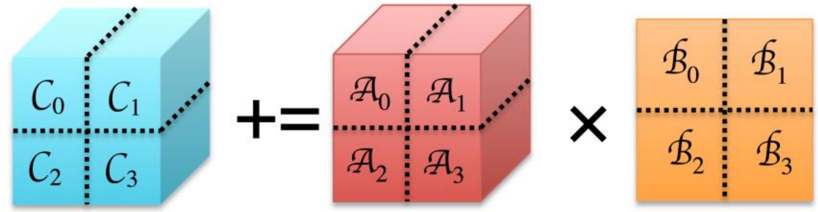




Strassen's Algorithm for Tensor Contraction

Jianyu Huang, Devin A. Matthews, Robert A. van de Geijn

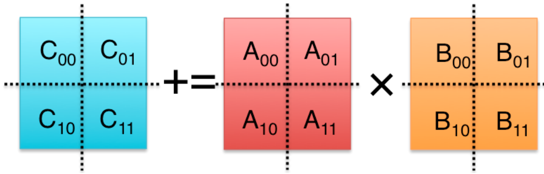
The University of Texas at Austin

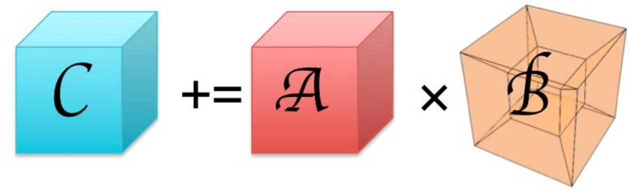
September 14-15, 2017

Tensor Computation Workshop

Flatiron Institute, 162 Fifth Ave, New York City.

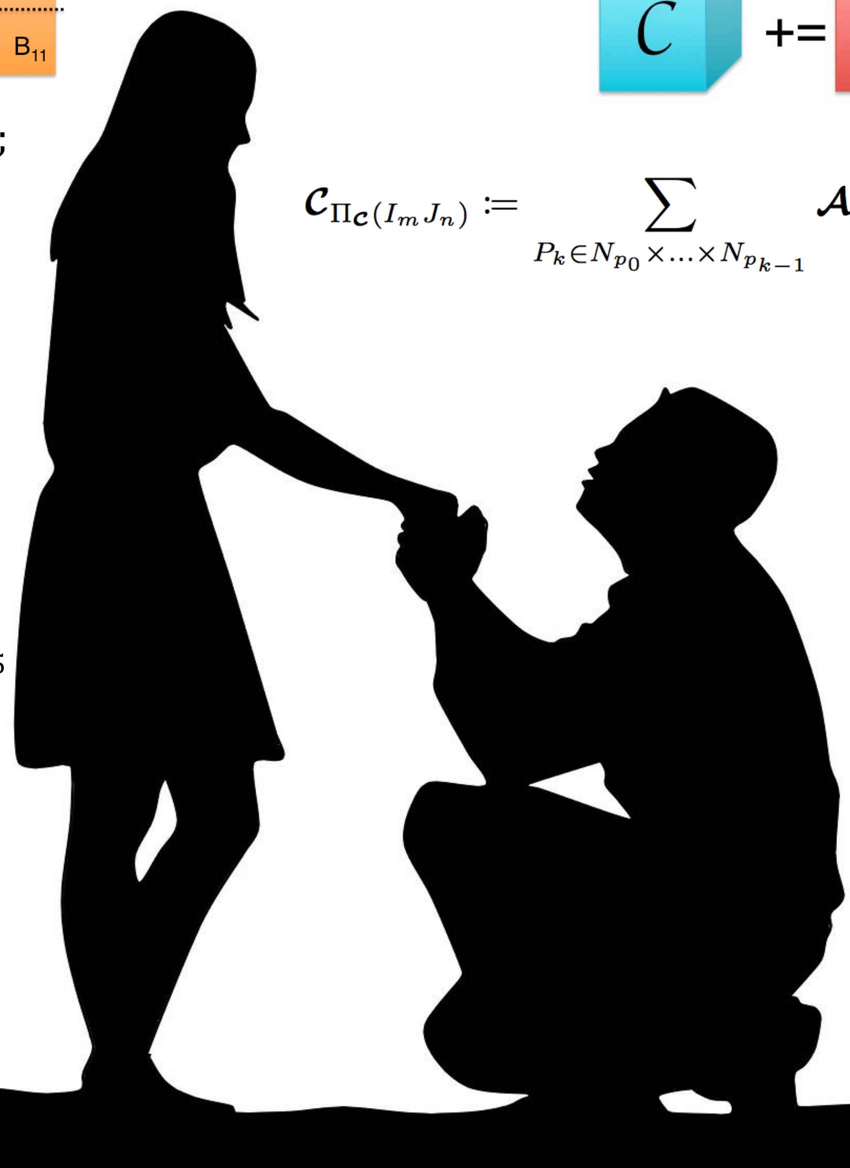
Marry Strassen with Tensor Contraction ?





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

$$\mathcal{C}_{\Pi_{\mathcal{C}}(I_m J_n)} := \sum_{P_k \in N_{p_0} \times \ldots \times N_{p_{k-1}}} \mathcal{A}_{\Pi_{\mathcal{A}}(I_m P_k)} \cdot \mathcal{B}_{\Pi_{\mathcal{B}}(P_k J_n)} + \mathcal{C}_{\Pi_{\mathcal{C}}(I_m J_n)}$$

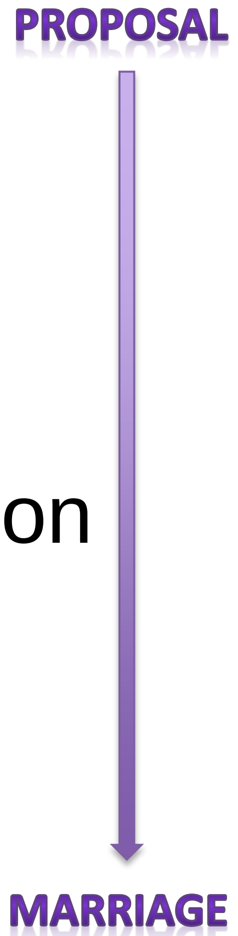


Practical Speedup?

$O(n^3) \rightarrow O(n^{2.8})$

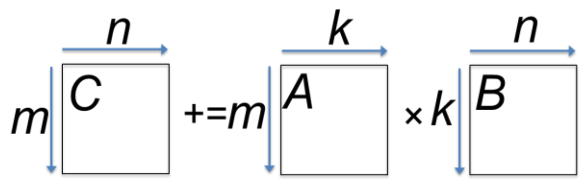
Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
  - High-performance Tensor Contraction
- Strassen's Algorithm for Tensor Contraction
- Performance Model
- Experiments
- Conclusion

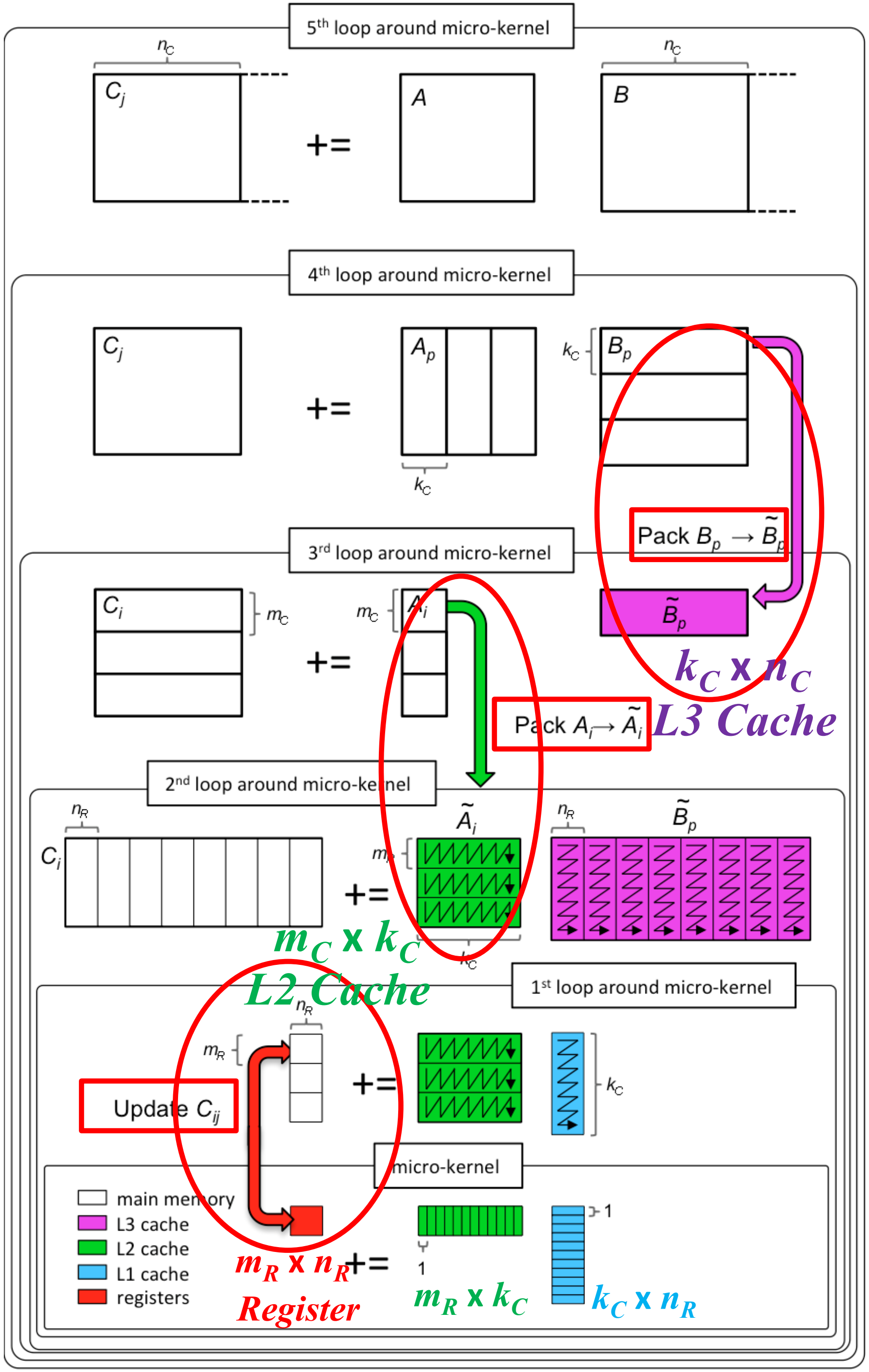


High-performance matrix multiplication (GEMM)

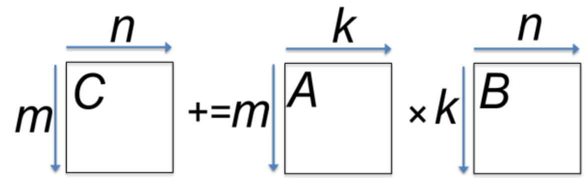
State-of-the-art GEMM in BLIS



- 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

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

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

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

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

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

// 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.*

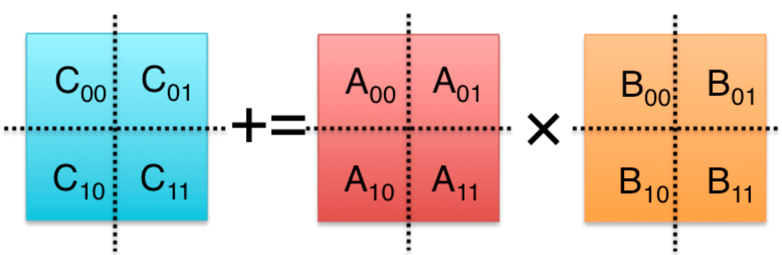
High-performance Strassen

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

Strassen's Algorithm Reloaded

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

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



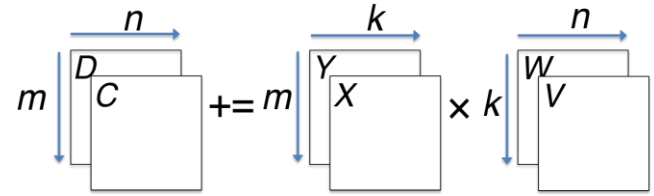
$$\boxed{M := (X+Y)(V+W); \qquad C \mathrel{+}= M; \qquad D \mathrel{+}= M;}$$

General operation for one-level Strassen:

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

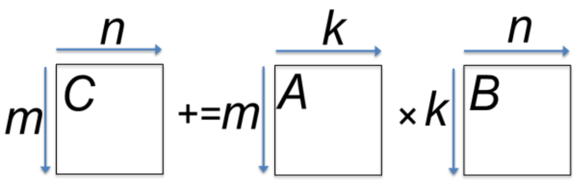
*__Jianyu Huang__, Tyler Smith, Greg Henry, and Robert van de Geijn. "Strassen's Algorithm Reloaded." In *SC'16*.

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

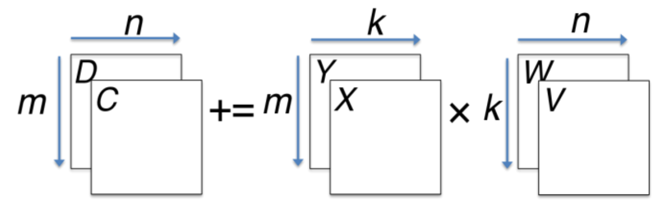


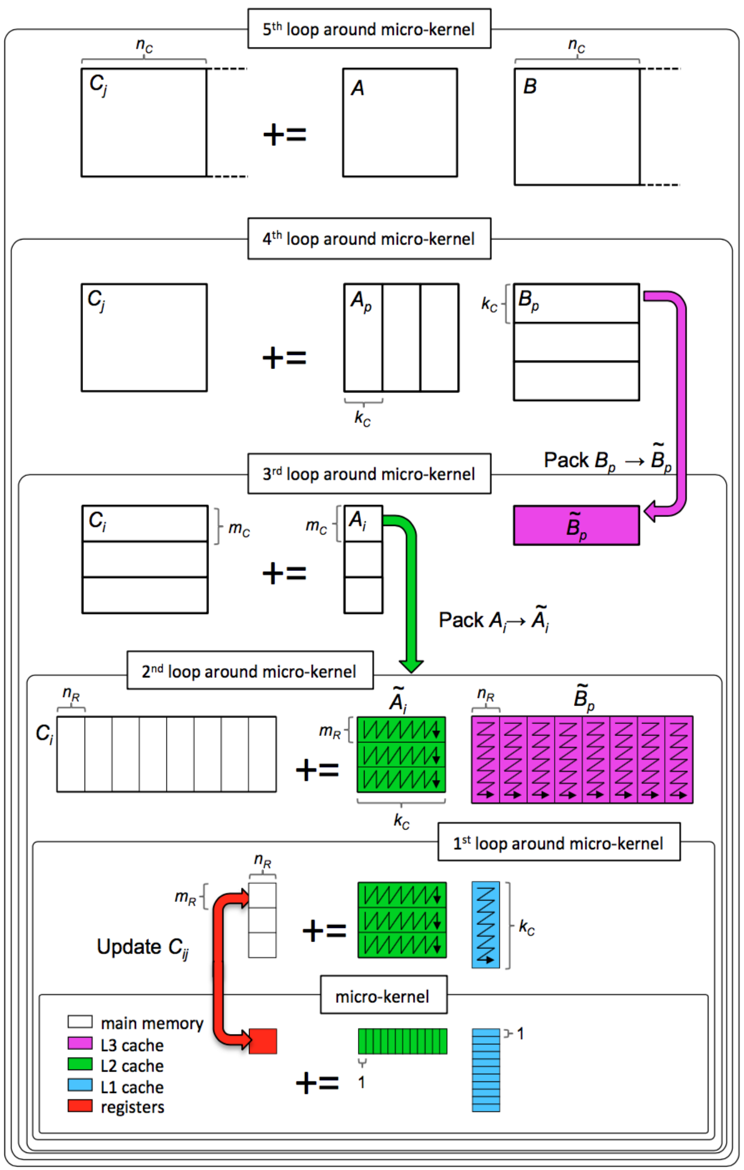
*__Jianyu Huang__, Tyler Smith, Greg Henry, and Robert van de Geijn. "Strassen's Algorithm Reloaded." In *SC'16*.

$C += AB;$

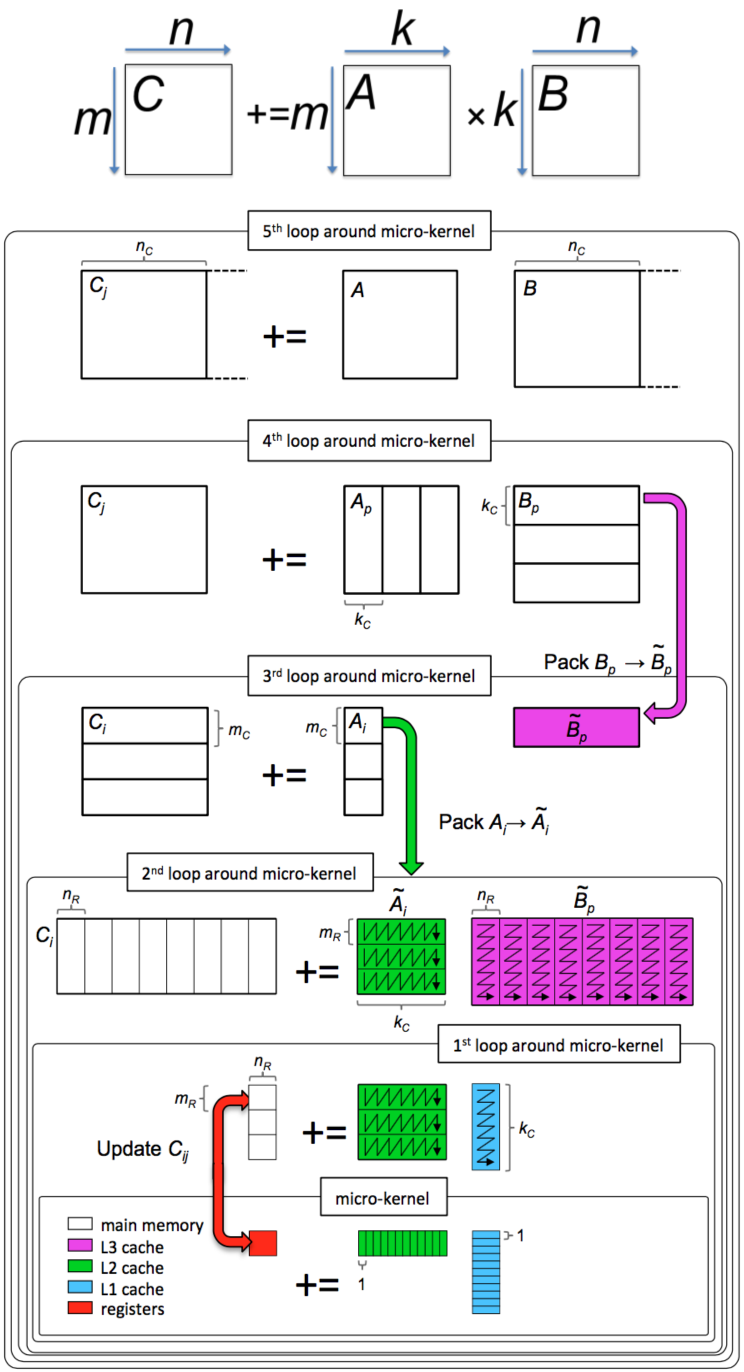


$M := (X+Y)(V+W);\ \ C += M;\ D += M;$

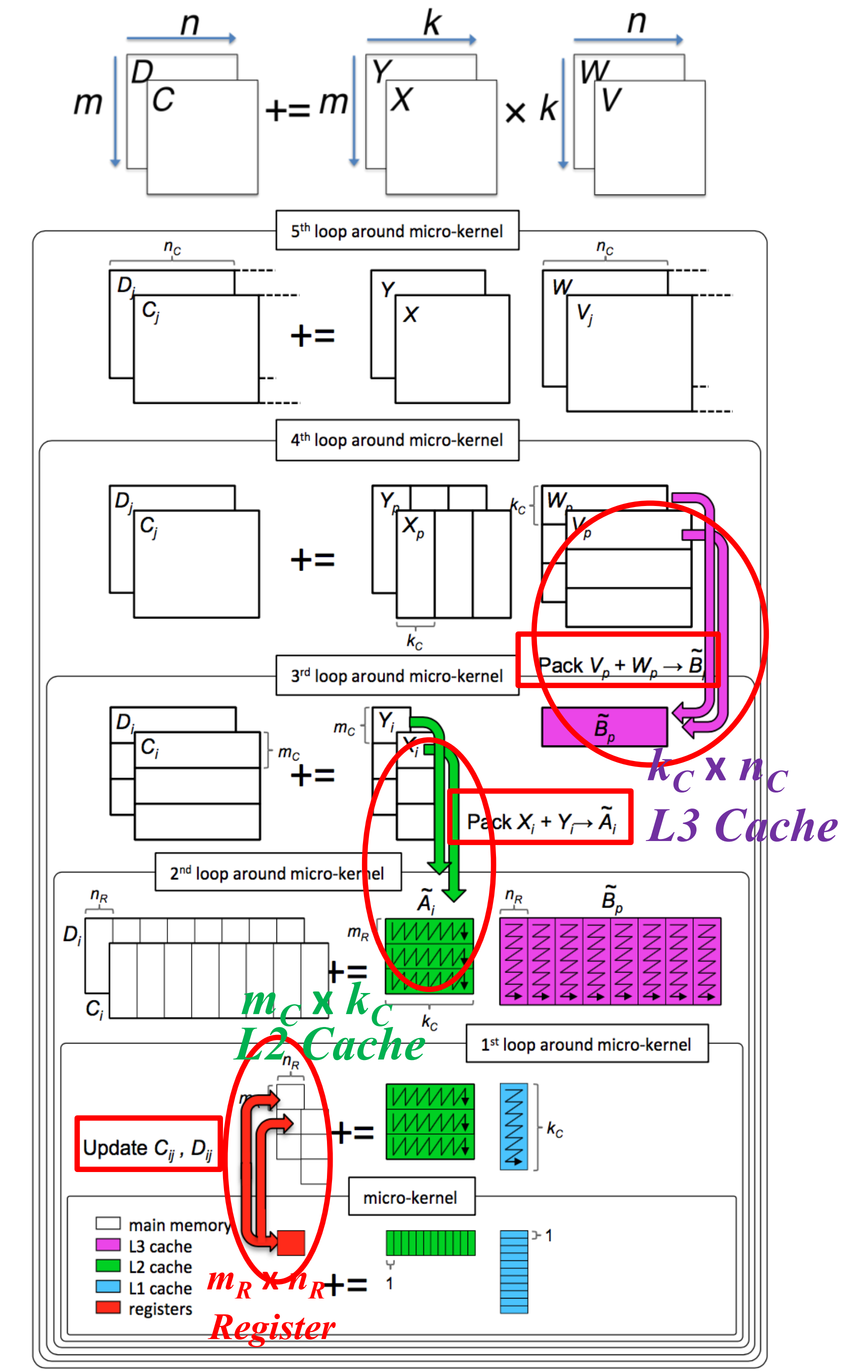




$C += AB;$



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

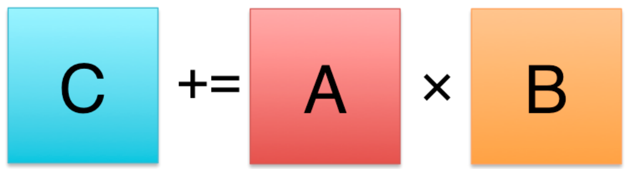


High-performance Tensor Contraction

Devin A. Matthews. "High-Performance Tensor Contraction without Transposition." Accepted in *SISC*.

Matrix vs. Tensor

Matrix Multiplication

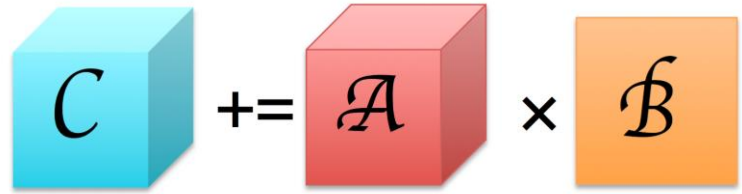


$$C += AB$$

$$C_{i,j} += \sum_k A_{i,k} B_{k,j}$$

BLAS/BLIS!

Tensor Contraction



$$\boldsymbol{\mathcal{C}} += \boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{B}}$$

$$\boldsymbol{\mathcal{C}}_{a,b,c} += \sum_d \boldsymbol{\mathcal{A}}_{d,c,a} \boldsymbol{\mathcal{B}}_{d,b}$$

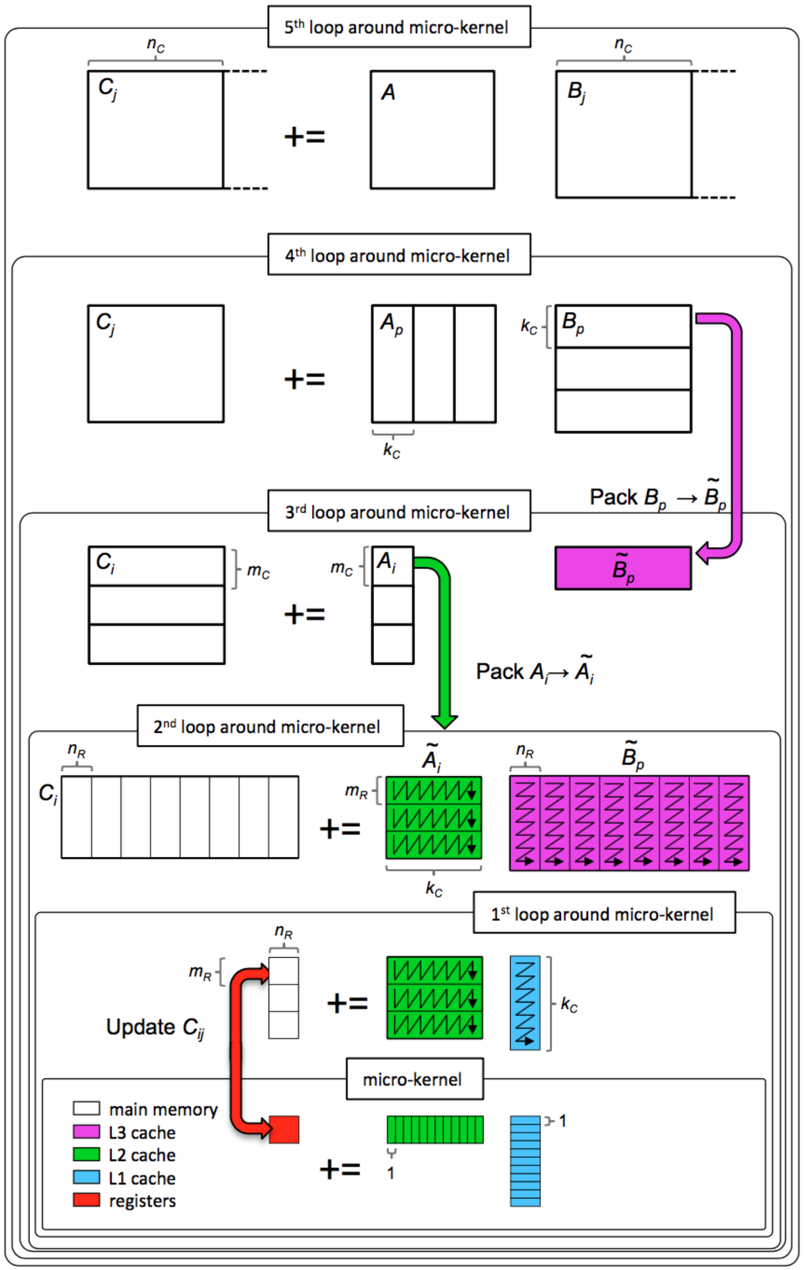
TBLIS/GETT

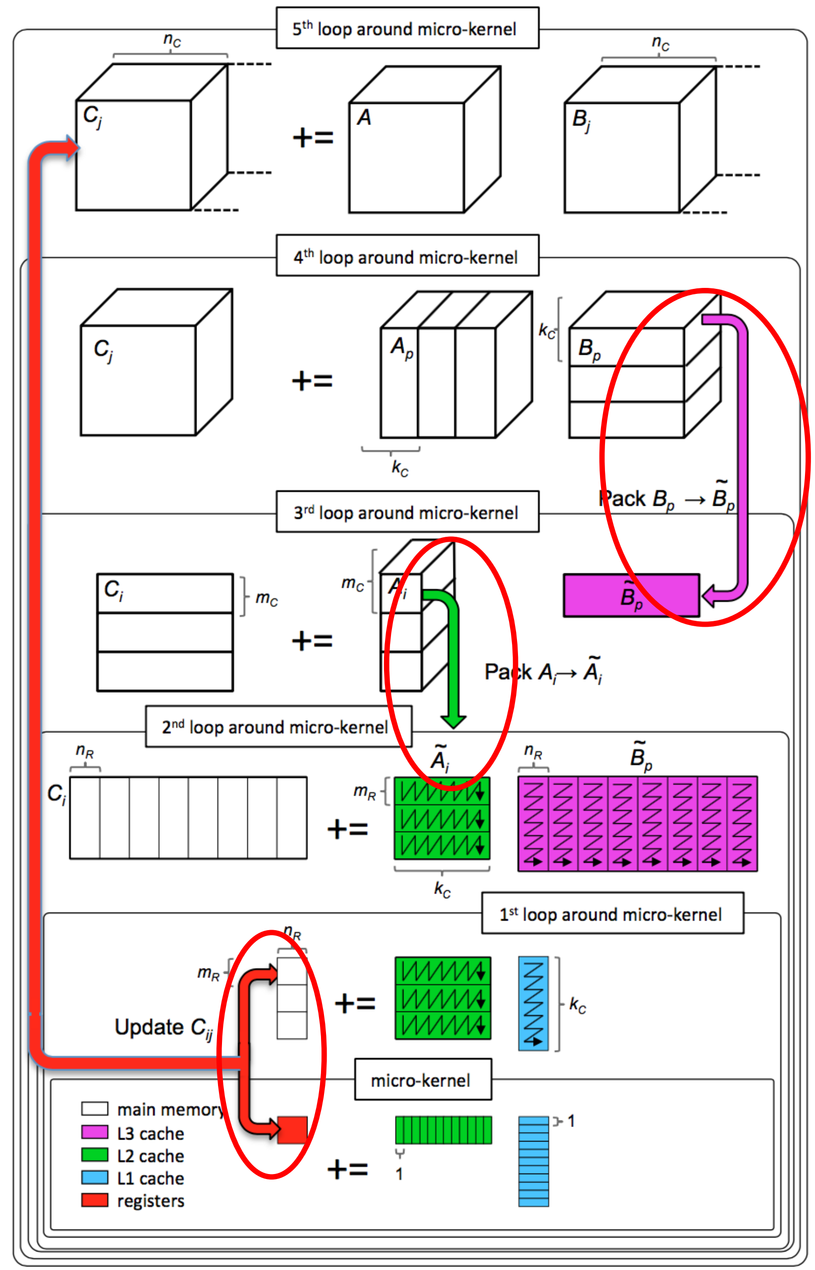
Paul Springer, and Paolo Bientinesi. "Design of a high-performance GEMM-like tensor-tensor multiplication." arXiv preprint arXiv:1607.00145 (2016).

Devin A. Matthews. “High-Performance Tensor Contraction without Transposition.” Accepted in *SISC*.

$$C := AB + C$$

$$\mathcal{C}_{\Pi_{\mathcal{C}}(I_m J_n)} := \sum_{P_k \in N_{p_0} \times \ldots \times N_{p_{k-1}}} \mathcal{A}_{\Pi_{\mathcal{A}}(I_m P_k)} \cdot \mathcal{B}_{\Pi_{\mathcal{B}}(P_k J_n)} + \mathcal{C}_{\Pi_{\mathcal{C}}(I_m J_n)}$$



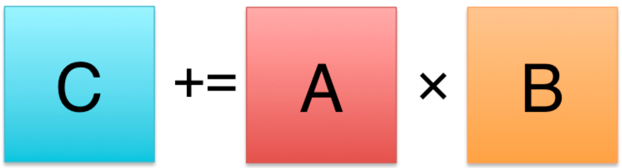


Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
  - High-performance Tensor Contraction
- Strassen's Algorithm for Tensor Contraction
- Performance Model
- Experiments
- Conclusion

Matrix vs. Tensor

Matrix Multiplication

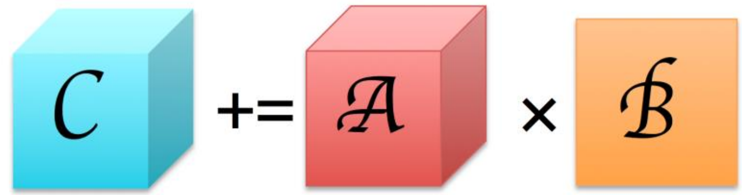


$$C += AB$$

$$C_{i,j} += \sum_k A_{i,k} B_{k,j}$$

BLAS/BLIS!

Tensor Contraction



$$\mathcal{C} += \mathcal{A}\mathcal{B}$$

$$\mathcal{C}_{a,b,c} += \sum_d \mathcal{A}_{d,c,a} \mathcal{B}_{d,b}$$

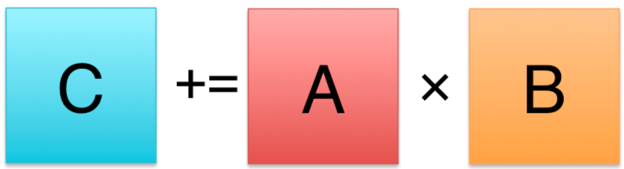
TBLIS/GETT

Paul Springer, and Paolo Bientinesi. "Design of a high-performance GEMM-like tensor-tensor multiplication." arXiv preprint arXiv:1607.00145 (2016).

Devin A. Matthews. “High-Performance Tensor Contraction without Transposition.” Accepted in *SISC*.

Matrix vs. Tensor

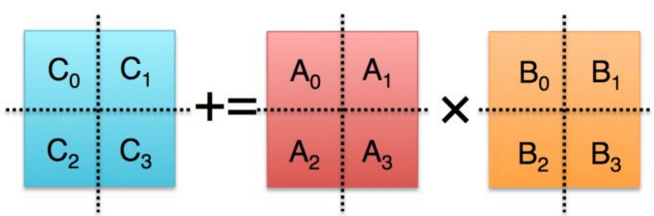
Matrix Multiplication



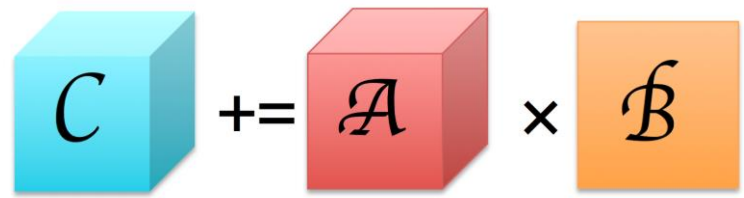
$$C += AB$$

$$C_{i,j} += \sum_k A_{i,k} B_{k,j}$$

BLAS/BLIS!



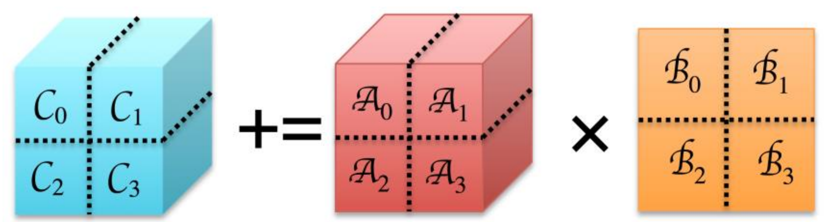
Tensor Contraction



$$\mathcal{C} += \mathcal{A}\mathcal{B}$$

$$\mathcal{C}_{a,b,c} += \sum_d \mathcal{A}_{d,c,a} \mathcal{B}_{d,b}$$

TBLIS/GETT

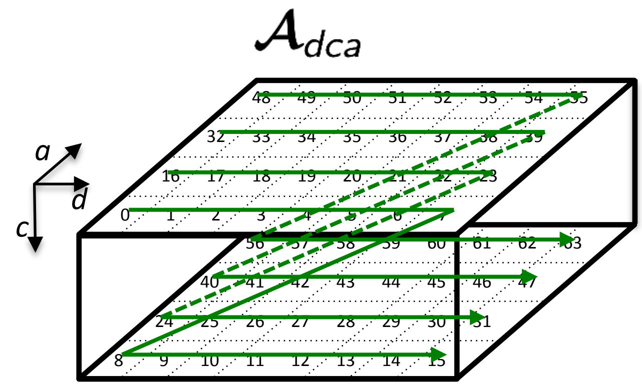


Tensors As Matrices: Block-Scatter-Matrix View

- **Tensor**: $\mathcal{A}_{dca}$, 8x2x4

with $N_a = 4, N_c = 2, N_d = 8$

- "d" dimension is stride-1, other dimensions have increasing strides (8, 16).



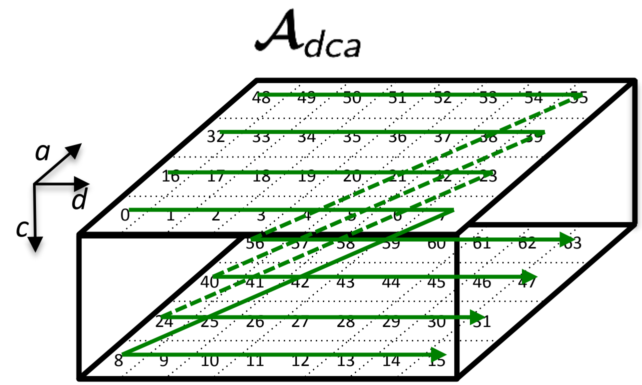
Devin A. Matthews. "High-Performance Tensor Contraction without Transposition." Accepted in *SISC*.

Tensors As Matrices: Block-Scatter-Matrix View

- **Tensor**: $\mathcal{A}_{dca}$, 8x2x4

with $N_a = 4, N_c = 2, N_d = 8$

- ❑ "d" dimension is stride-1, other dimensions have increasing strides (8, 16).



- **Matrix**: $A_{(ac)(d)}$, 8x8

with $N_{I_m} = N_a \cdot N_c = 8, N_{P_k} = N_d = 8$

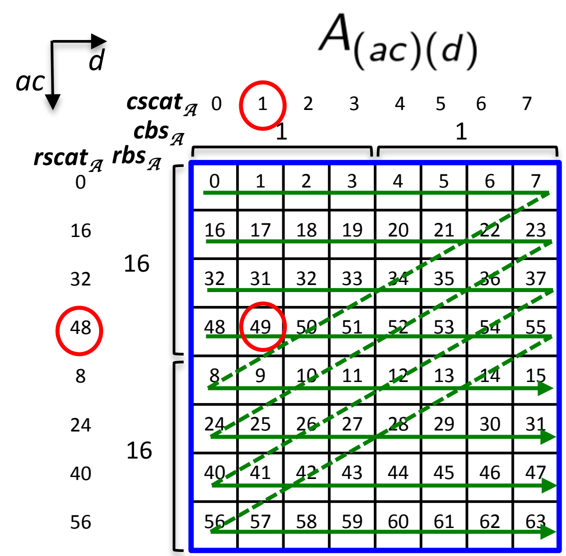
- ❑ Column "ac" dimension has stride of "c" (8x2=16).

Row "d" dimension has is stirde-1. (i.e. A is row-major.)

- ❑ $rscat_{\mathcal{A}}$, $cscat_{\mathcal{A}}$ store offset for each position in rows or columns: Scatter-Matrix Vector

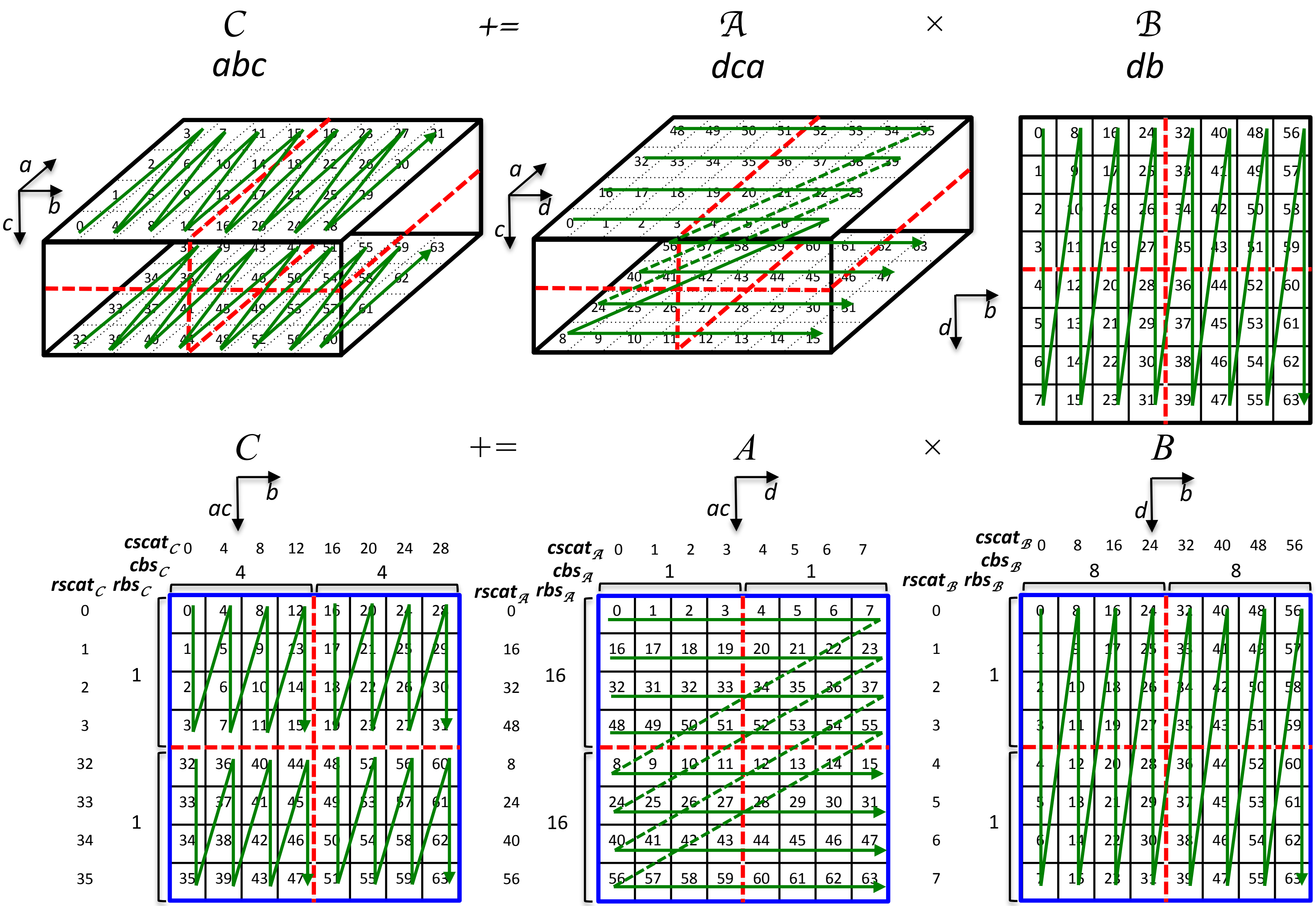
$$\mathrm{OFFSET}_{\mathcal{A}_{d,c,a}} = rscat_{A;(ac)} + cscat_{A;(d)}$$

- ❑ $rbs_{\mathcal{A}}$, $cbs_{\mathcal{A}}$ store stride for each block or zero for irregular blocks: Block-Scatter-Matrix Vector
  - vector load/store instructions for stride-1 index
  - vector gather/scatter instructions for stride-n index.



Devin A. Matthews. "High-Performance Tensor Contraction without Transposition." Accepted in *SISC*.

Strassen's Algorithm for Tensor Contraction



Jianyu Huang, Devin A. Matthews, and Robert A. van de Geijn. "Strassen's Algorithm for Tensor Contraction." arXiv:1704.03092 (2017).

Modifications to GEMM

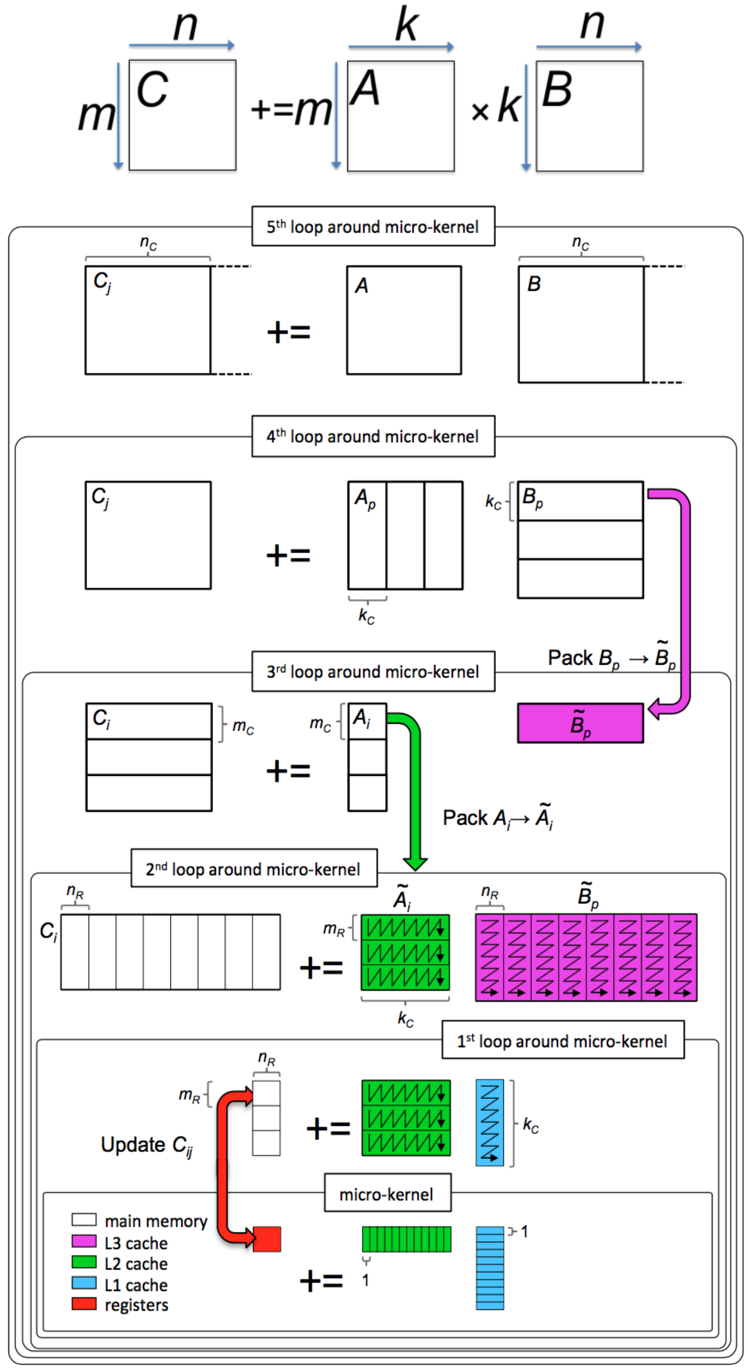
$$M_0 := (A_{00}+A_{11})(B_{00}+B_{11}); \quad C_{00} += M_0; C_{11} += M_0;$$

- Packing routines:
  - Implicit tensor-to-matrix permutations
  - Addition of submatrices of A and B.
- Micro-kernel:
  - Implicit matrix-to-tensor transformations
  - Scatter update of submatrices of C.

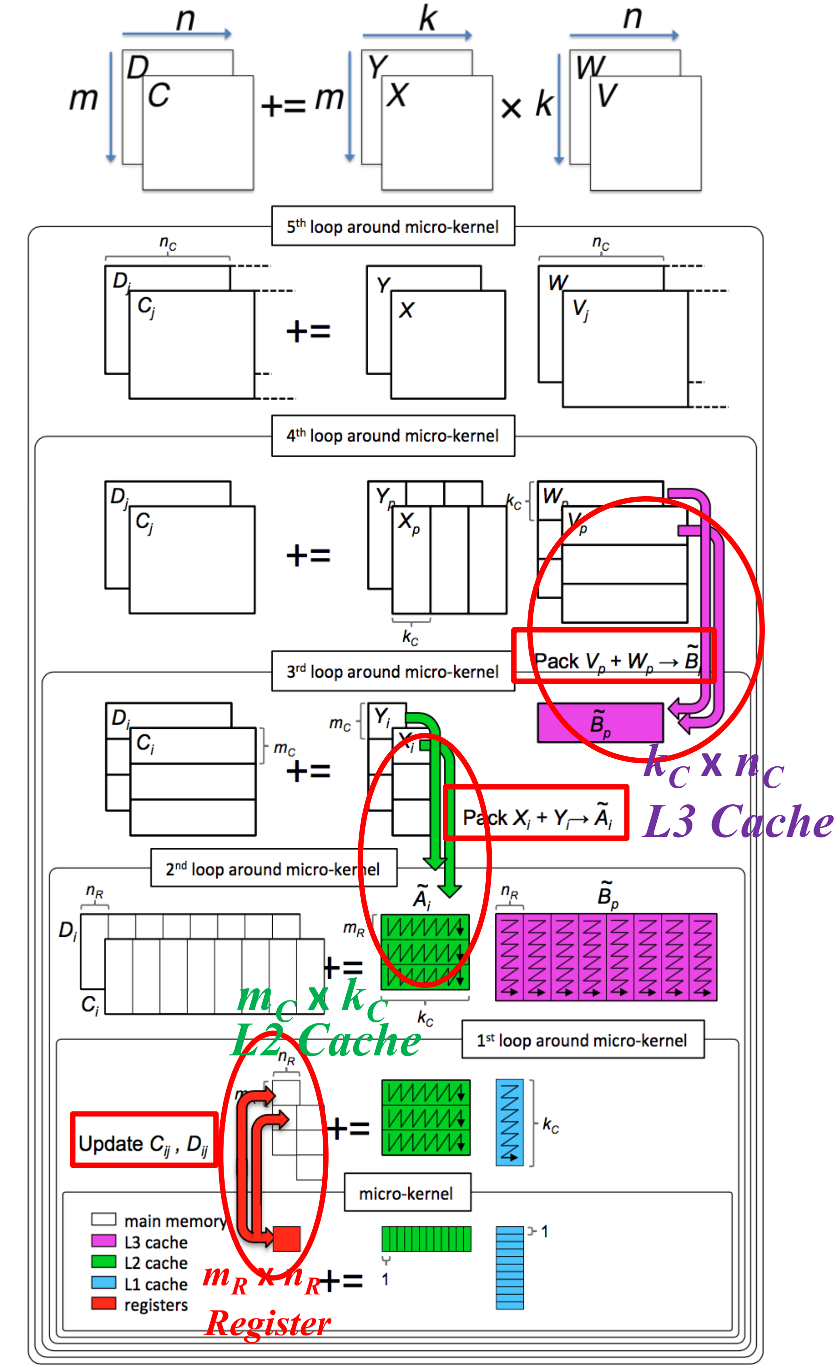
❑ ~~Additional workspace for Transposition (Tensor Contraction)~~

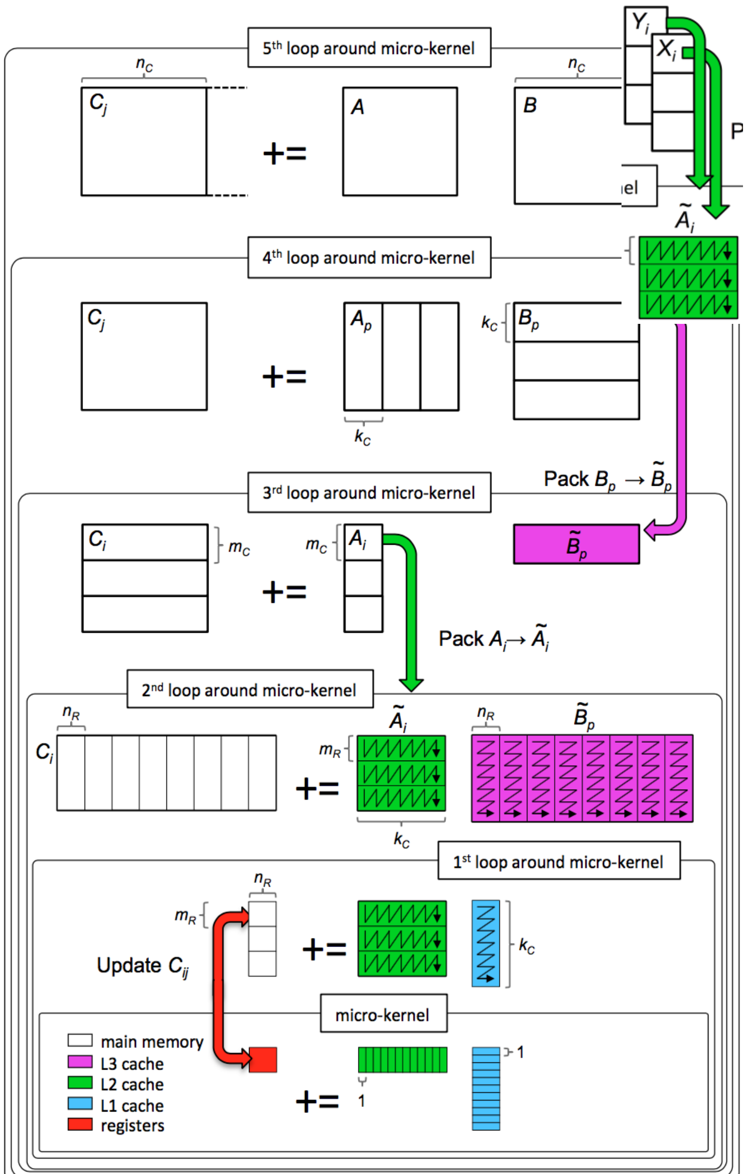
❑ ~~Additional Workspace for Summation (Strassen)~~

$C += AB;$

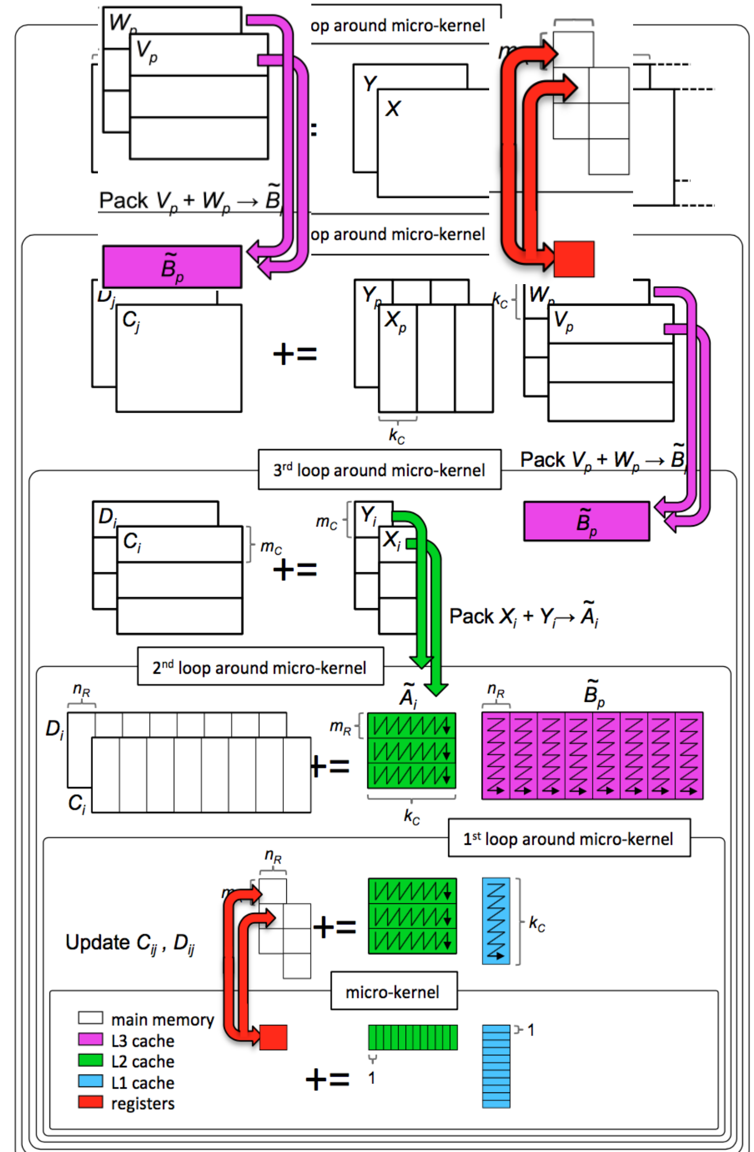


$M := (X+Y)(V+W); \; C += M; \; D += M;$



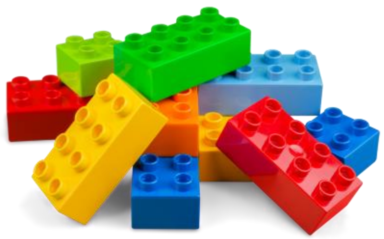


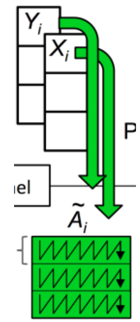
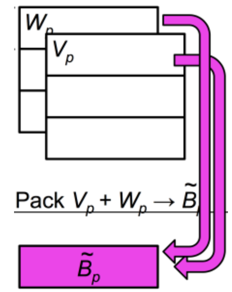
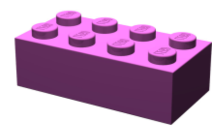
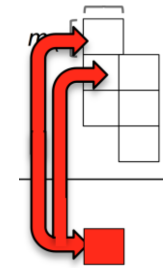
5th loop around micro-kernel
n_C
C_j
+=
A
B
Y_i
X_i
$\tilde{A}_i$
4th loop around micro-kernel
C_j
A_p
B_p
k_C
Pack $B_p \rightarrow \tilde{B}_p$
3rd loop around micro-kernel
C_i
m_C
A_i
$\tilde{B}_p$
Pack $A_i \rightarrow \tilde{A}_i$
2nd loop around micro-kernel
n_R
m_R
1st loop around micro-kernel
Update C_{ij}
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
1



op around micro-kernel
W_p
V_p
Y
X
Pack $V_p + W_p \rightarrow \tilde{B}$
$\tilde{B}_p$
D_j
C_j
Y_p
X_p
k_C
Pack $V_p + W_p \rightarrow \tilde{B}$
3rd loop around micro-kernel
D_i
C_i
m_C
Y_i
X_i
Pack $X_i + Y_i \rightarrow \tilde{A}_i$
2nd loop around micro-kernel
$\tilde{A}_i$
$\tilde{B}_p$
n_R
m_R
1st loop around micro-kernel
Update C_{ij} , D_{ij}
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
1

Variations on a theme



	 	 	 
• Naïve Strassen	✘	✘	✘
• AB Strassen	✔	✔	✘
• ABC Strassen	✔	✔	✔

Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
  - High-performance Tensor Contraction
- Strassen’s Algorithm for Tensor Contraction
- Performance Model
- Experiments
- Conclusion

Performance Model

- Performance Metric

$$\boxed{\mathcal{C}_{\Pi_{\mathcal{C}}(I_m J_n)} := \sum_{P_k \in N_{p_0} \times \ldots \times N_{p_{k-1}}} \mathcal{A}_{\Pi_{\mathcal{A}}(I_m P_k)} \cdot \mathcal{B}_{\Pi_{\mathcal{B}}(P_k J_n)} + \mathcal{C}_{\Pi_{\mathcal{C}}(I_m J_n)}}$$

$$N_{I_m} = \prod_{i \in I_m} N_i = N_{i_0} \cdot \ldots \cdot N_{i_{m-1}},$$
$$N_{J_n} = \prod_{j \in J_n} N_j = N_{j_0} \cdot \ldots \cdot N_{j_{n-1}},$$
$$N_{P_k} = \prod_{p \in P_k} N_p = N_{p_0} \cdot \ldots \cdot N_{p_{k-1}}.$$

$$\textit{Effective}\ \text{GFLOPS} = \frac{2 \cdot N_{I_m} \cdot N_{J_n} \cdot N_{P_k}}{\text{T (in seconds)}} \cdot 10^{-9}$$

- Total Time Breakdown

$$T = T_a + T_m$$

Arithmetic Operations

Memory Operations

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

	type	τ	TBLIS	L-level
$T_a^{\times}$	-	τ_a	$2N_{I_m}N_{J_n}N_{P_k}$	$2\frac{N_{I_m}}{2^L}\frac{N_{J_n}}{2^L}\frac{N_{P_k}}{2^L}$
$T_a^{\mathcal{A}+}$	-	τ_a	-	$2\frac{N_{I_m}}{2^L}\frac{N_{P_k}}{2^L}$
$T_a^{\mathcal{B}+}$	-	τ_a	-	$2\frac{N_{P_k}}{2^L}\frac{N_{J_n}}{2^L}$
$T_a^{\mathcal{C}+}$	-	τ_a	-	$2\frac{N_{I_m}}{2^L}\frac{N_{J_n}}{2^L}$

	TBLIS	1-level			2-level		
		ABC	AB	Naive	ABC	AB	Naive
$W_a^{\times}$	1	7	7	7	49	49	49
$W_a^{\mathcal{A}+}$	-	5	5	5	95	95	95
$W_a^{\mathcal{B}+}$	-	5	5	5	95	95	95
$W_a^{\mathcal{C}+}$	-	12	12	12	144	144	144

$$N_{I_m} = \prod_{i \in I_m} N_i = N_{i_0} \cdot \ldots \cdot N_{i_{m-1}},$$
$$N_{J_n} = \prod_{j \in J_n} N_j = N_{j_0} \cdot \ldots \cdot N_{j_{n-1}},$$
$$N_{P_k} = \prod_{p \in P_k} N_p = N_{p_0} \cdot \ldots \cdot N_{p_{k-1}}.$$

$$T_a = W_a^{\times} \cdot T_a^{\times} + W_a^{\mathcal{A}+} \cdot T_a^{\mathcal{A}+} + W_a^{\mathcal{B}+} \cdot T_a^{\mathcal{B}+} + W_a^{\mathcal{C}+} \cdot T_a^{\mathcal{C}+}$$
$$T_m = W_m^{\mathcal{A}\times} \cdot T_m^{\mathcal{A}\times} + W_m^{\mathcal{B}\times} \cdot T_m^{\mathcal{B}\times} + W_m^{\mathcal{C}\times} \cdot T_m^{\mathcal{C}\times} + W_m^{\mathcal{A}+} \cdot T_m^{\mathcal{A}+} + W_m^{\mathcal{B}+} \cdot T_m^{\mathcal{B}+} + W_m^{\mathcal{C}+} \cdot T_m^{\mathcal{C}+}$$

	type	τ	TBLIS	L-level
$T_a^{\times}$	-	τ_a	$2N_{I_m}N_{J_n}N_{P_k}$	$2\frac{N_{I_m}}{2^L}\frac{N_{J_n}}{2^L}\frac{N_{P_k}}{2^L}$
$T_a^{\mathcal{A}+}$	-	τ_a	-	$2\frac{N_{I_m}}{2^L}\frac{N_{P_k}}{2^L}$
$T_a^{\mathcal{B}+}$	-	τ_a	-	$2\frac{N_{P_k}}{2^L}\frac{N_{J_n}}{2^L}$
$T_a^{\mathcal{C}+}$	-	τ_a	-	$2\frac{N_{I_m}}{2^L}\frac{N_{J_n}}{2^L}$
$T_m^{\mathcal{A}\times}$	`r`	τ_b	$N_{I_m}N_{P_k}\lceil\frac{N_{J_n}}{n_c}\rceil$	$\frac{N_{I_m}}{2^L}\frac{N_{P_k}}{2^L}\lceil\frac{N_{J_n}/2^L}{n_c}\rceil$
$T_m^{\widetilde{A}\times}$	`w`	τ_b	$N_{I_m}N_{P_k}\lceil\frac{N_{J_n}}{n_c}\rceil$	$\frac{N_{I_m}}{2^L}\frac{N_{P_k}}{2^L}\lceil\frac{N_{J_n}/2^L}{n_c}\rceil$
$T_m^{\mathcal{B}\times}$	`r`	τ_b	$N_{J_n}N_{P_k}$	$\frac{N_{J_n}}{2^L}\frac{N_{P_k}}{2^L}$
$T_m^{\widetilde{B}\times}$	`w`	τ_b	$N_{J_n}N_{P_k}$	$\frac{N_{J_n}}{2^L}\frac{N_{P_k}}{2^L}$
$T_m^{\mathcal{C}\times}$	`r/w`	τ_b	$2\lambda N_{I_m}N_{J_n}\lceil\frac{N_{P_k}}{k_c}\rceil$	$2\lambda\frac{N_{I_m}}{2^L}\frac{N_{J_n}}{2^L}\lceil\frac{N_{P_k}/2^L}{k_c}\rceil$
$T_m^{\mathcal{A}+}$	`r/w`	τ_b	$N_{I_m}N_{P_k}$	$\frac{N_{I_m}}{2^L}\frac{N_{P_k}}{2^L}$
$T_m^{\mathcal{B}+}$	`r/w`	τ_b	$N_{J_n}N_{P_k}$	$\frac{N_{J_n}}{2^L}\frac{N_{P_k}}{2^L}$
$T_m^{\mathcal{C}+}$	`r/w`	τ_b	$N_{I_m}N_{J_n}$	$\frac{N_{I_m}}{2^L}\frac{N_{J_n}}{2^L}$

	TBLIS	1-level			2-level		
		ABC	AB	Naive	ABC	AB	Naive
$W_a^{\times}$	1	7	7	7	49	49	49
$W_a^{\mathcal{A}+}$	-	5	5	5	95	95	95
$W_a^{\mathcal{B}+}$	-	5	5	5	95	95	95
$W_a^{\mathcal{C}+}$	-	12	12	12	144	144	144
$W_m^{\mathcal{A}\times}$	1	12	12	7	194	194	49
$W_m^{\widetilde{A}\times}$	-	-	-	-	-	-	-
$W_m^{\mathcal{B}\times}$	1	12	12	7	194	194	49
$W_m^{\widetilde{B}\times}$	-	-	-	-	-	-	-
$W_m^{\mathcal{C}\times}$	1	12	7	7	144	49	49
$W_m^{\mathcal{A}+}$	-	-	-	19	-	-	293
$W_m^{\mathcal{B}+}$	-	-	-	19	-	-	293
$W_m^{\mathcal{C}+}$	-	-	36	36	-	432	432

$$N_{I_m} = \prod_{i\in I_m} N_i = N_{i_0} \cdot \ldots \cdot N_{i_{m-1}},$$
$$N_{J_n} = \prod_{j\in J_n} N_j = N_{j_0} \cdot \ldots \cdot N_{j_{n-1}},$$
$$N_{P_k} = \prod_{p\in P_k} N_p = N_{p_0} \cdot \ldots \cdot N_{p_{k-1}}.$$

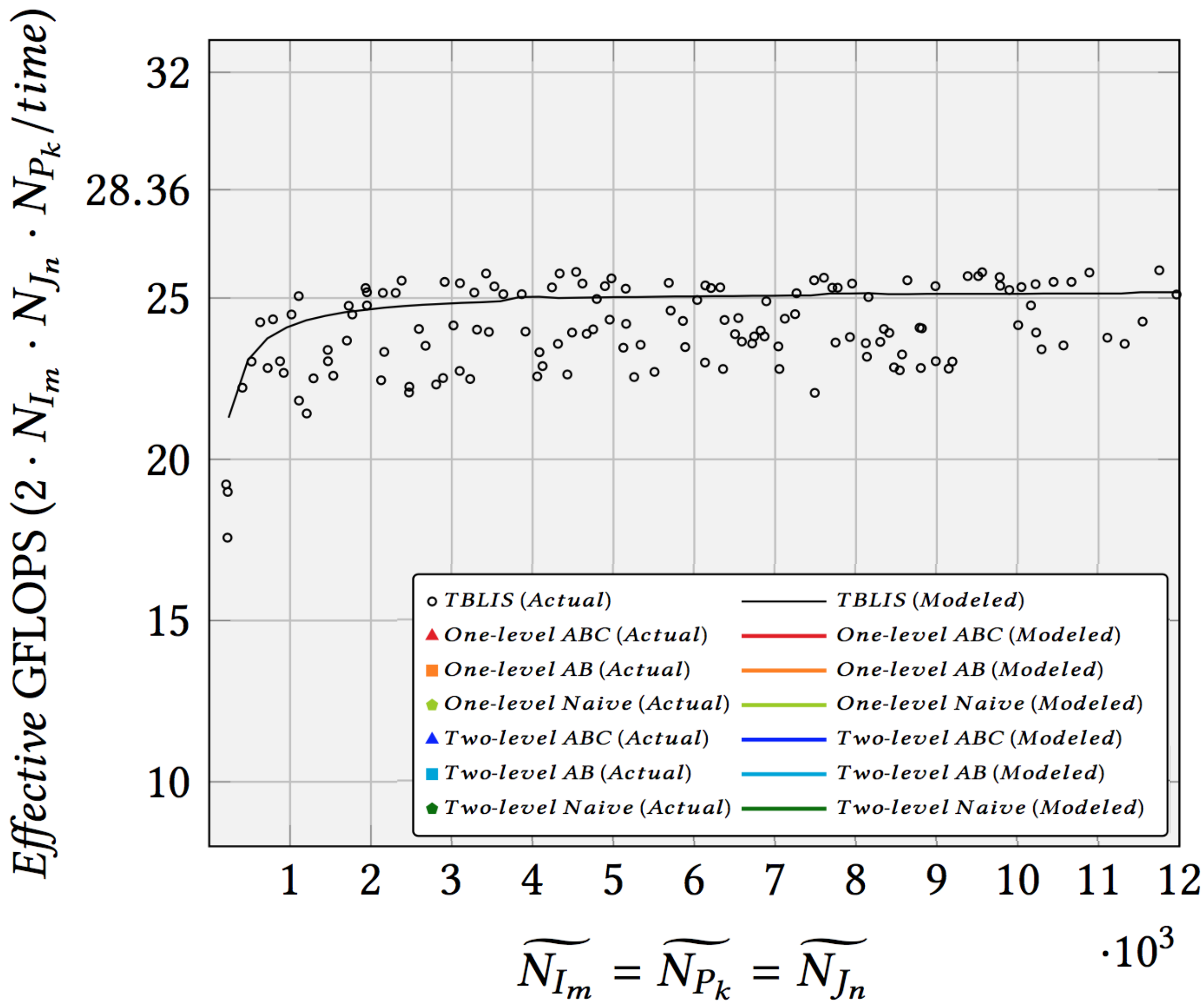
Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
  - High-performance Tensor Contraction
- Strassen's Algorithm for Tensor Contraction
- Performance Model
- Experiments
- Conclusion

Synthetic Tensor Contractions

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

$\widetilde{N_{I_m}} = \widetilde{N_{P_k}} = \widetilde{N_{J_n}}$, 1 core, *Actual vs. Modeled*

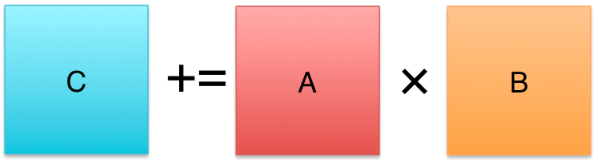


$$N_{I_m} = \prod_{i \in I_m} N_i = N_{i_0} \cdot \ldots \cdot N_{i_{m-1}},$$
$$N_{J_n} = \prod_{j \in J_n} N_j = N_{j_0} \cdot \ldots \cdot N_{j_{n-1}},$$
$$N_{P_k} = \prod_{p \in P_k} N_p = N_{p_0} \cdot \ldots \cdot N_{p_{k-1}}.$$

square:

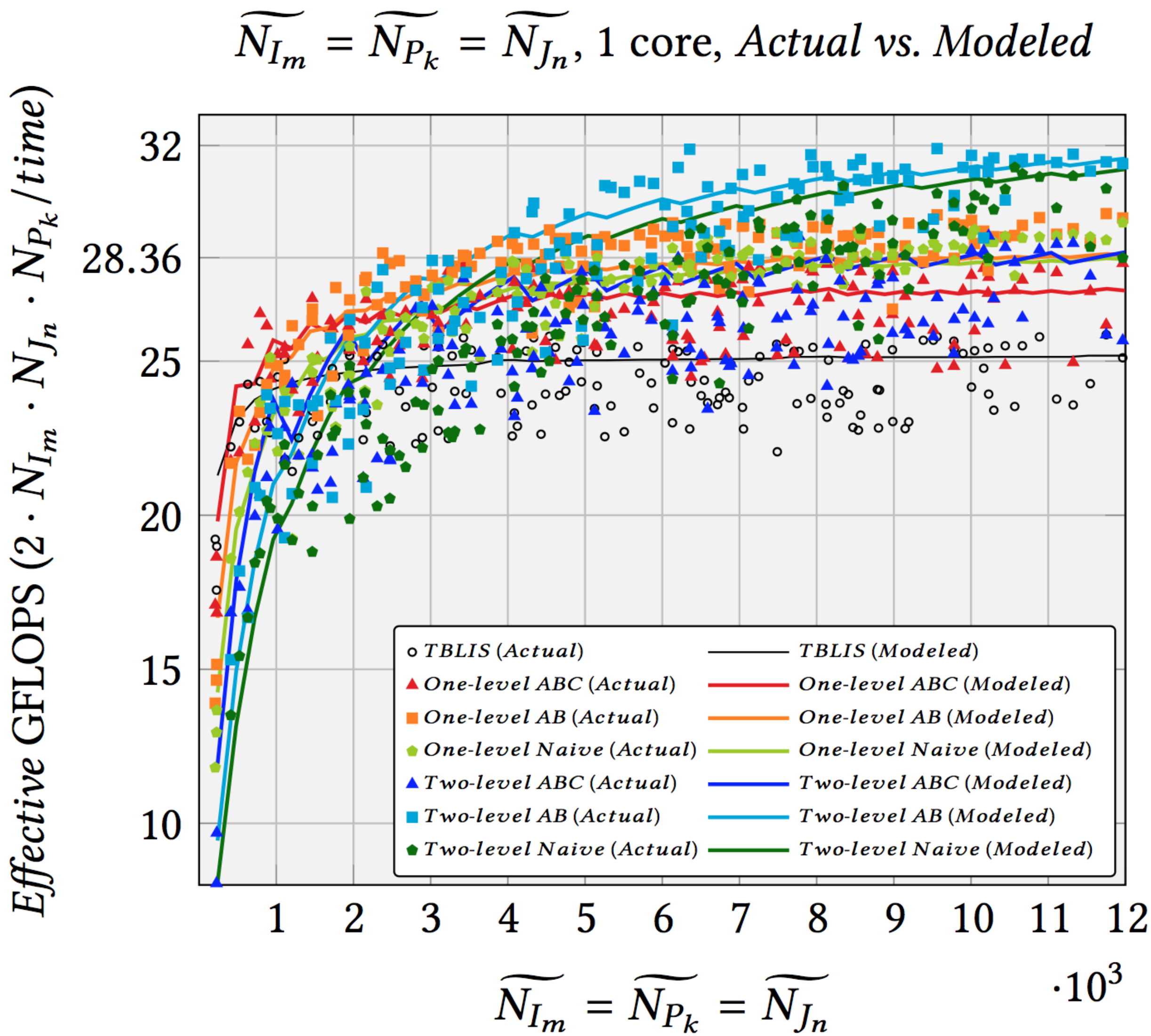
$$N_{I_m} \approx N_{J_n} \approx N_{P_k}$$

$$\widetilde{N_{I_m}} = \widetilde{N_{J_n}} = \widetilde{N_{P_k}} = (N_{I_m} \cdot N_{J_n} \cdot N_{P_k})^{1/3}.$$



Synthetic Tensor Contractions

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

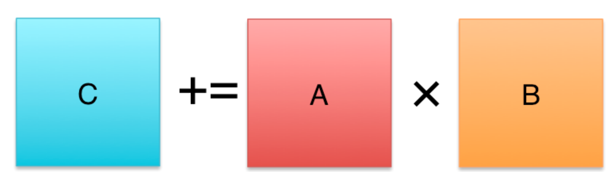


$$N_{I_m} = \prod_{i \in I_m} N_i = N_{i_0} \cdot \ldots \cdot N_{i_{m-1}},$$
$$N_{J_n} = \prod_{j \in J_n} N_j = N_{j_0} \cdot \ldots \cdot N_{j_{n-1}},$$
$$N_{P_k} = \prod_{p \in P_k} N_p = N_{p_0} \cdot \ldots \cdot N_{p_{k-1}}.$$

square:

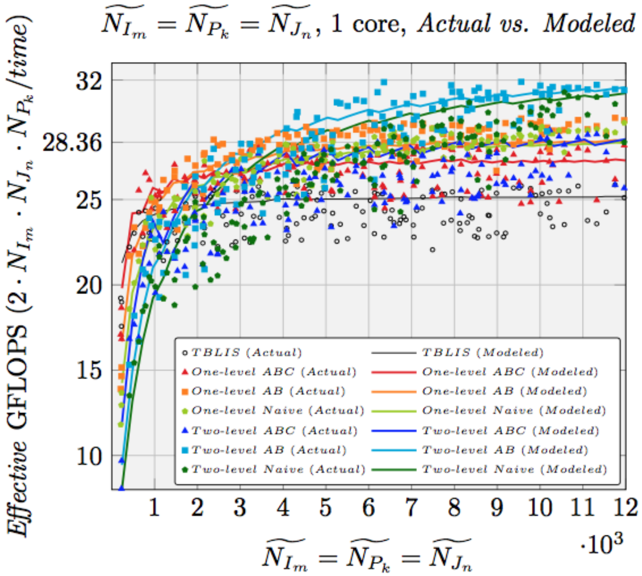
$$N_{I_m} \approx N_{J_n} \approx N_{P_k}$$

$$\widetilde{N_{I_m}} = \widetilde{N_{J_n}} = \widetilde{N_{P_k}} = (N_{I_m} \cdot N_{J_n} \cdot N_{P_k})^{1/3}.$$



Synthetic Tensor Contractions

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

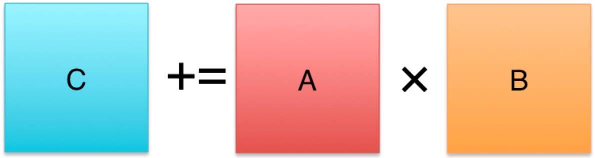


square:

$N_{I_m} \approx N_{J_n} \approx N_{P_k}$;

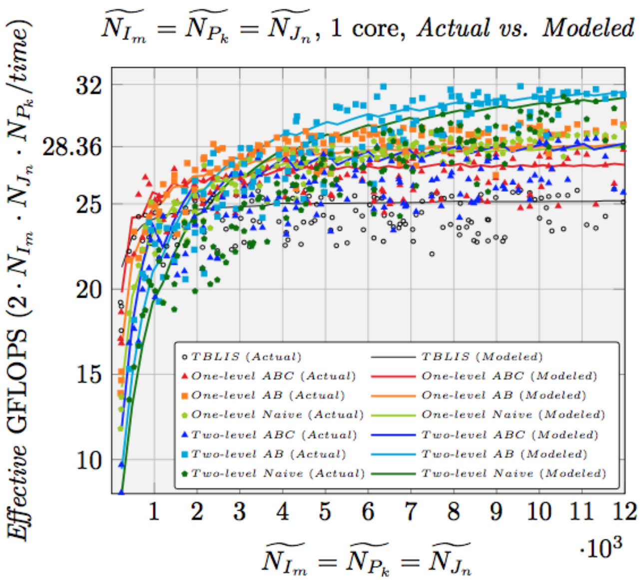
$\widetilde{N_{I_m}} = \widetilde{N_{J_n}} = \widetilde{N_{P_k}} = (N_{I_m} \cdot N_{J_n} \cdot N_{P_k})^{1/3}$.

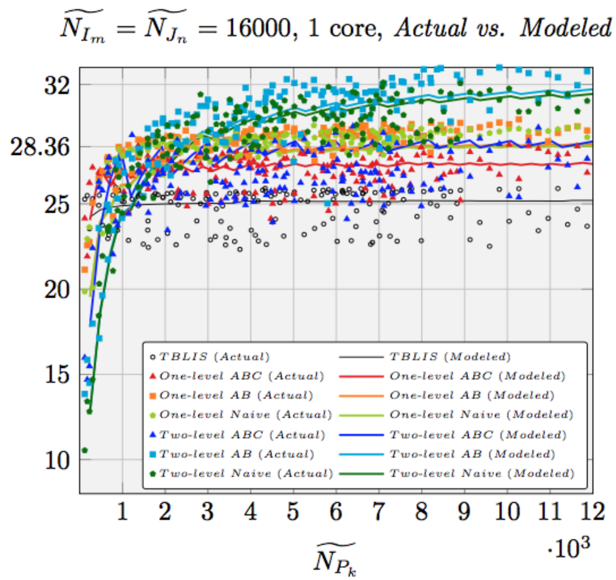
$$N_{I_m} = \prod_{i \in I_m} N_i = N_{i_0} \cdot \ldots \cdot N_{i_{m-1}},$$
$$N_{J_n} = \prod_{j \in J_n} N_j = N_{j_0} \cdot \ldots \cdot N_{j_{n-1}},$$
$$N_{P_k} = \prod_{p \in P_k} N_p = N_{p_0} \cdot \ldots \cdot N_{p_{k-1}}.$$

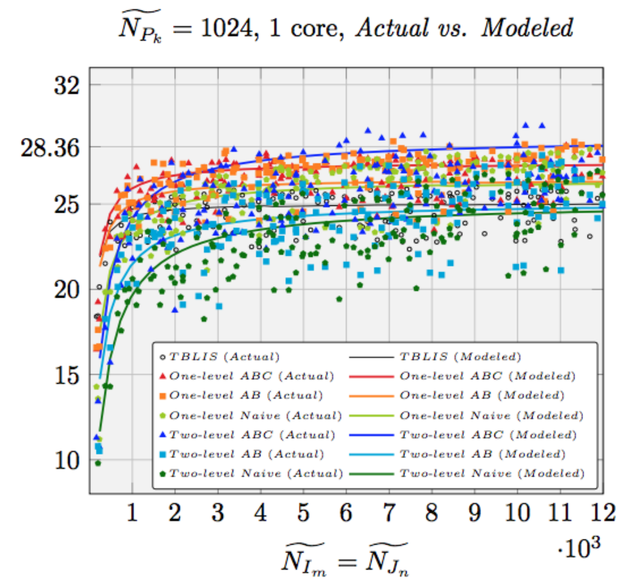


Synthetic Tensor Contractions

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





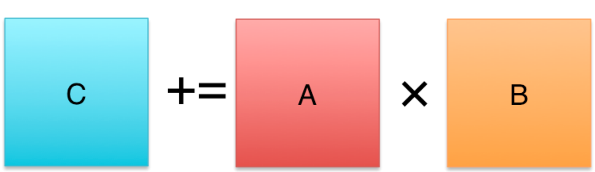


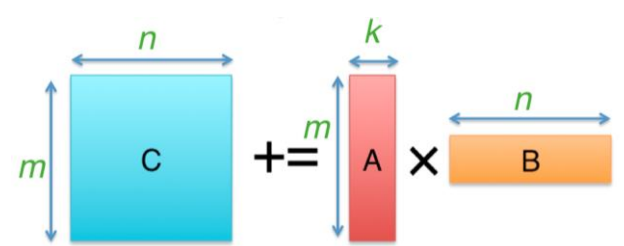
square:
$N_{I_m} \approx N_{J_n} \approx N_{P_k}$;
$\widetilde{N_{I_m}} = \widetilde{N_{J_n}} = \widetilde{N_{P_k}} = (N_{I_m} \cdot N_{J_n} \cdot N_{P_k})^{1/3}$.

rank-N_{P_k}:
$N_{I_m} \approx N_{J_n} \approx 16000$, N_{P_k} varies;
$\widetilde{N_{P_k}} = N_{I_m} \cdot N_{J_n} \cdot N_{P_k} / (16000 \cdot 16000)$.

fixed-N_{P_k}:
$N_{P_k} \approx 1024$, $N_{I_m} \approx N_{J_n}$ vary.
$\widetilde{N_{I_m}} = \widetilde{N_{J_n}} = (N_{I_m} \cdot N_{J_n} \cdot N_{P_k} / 1024)^{1/2}$.

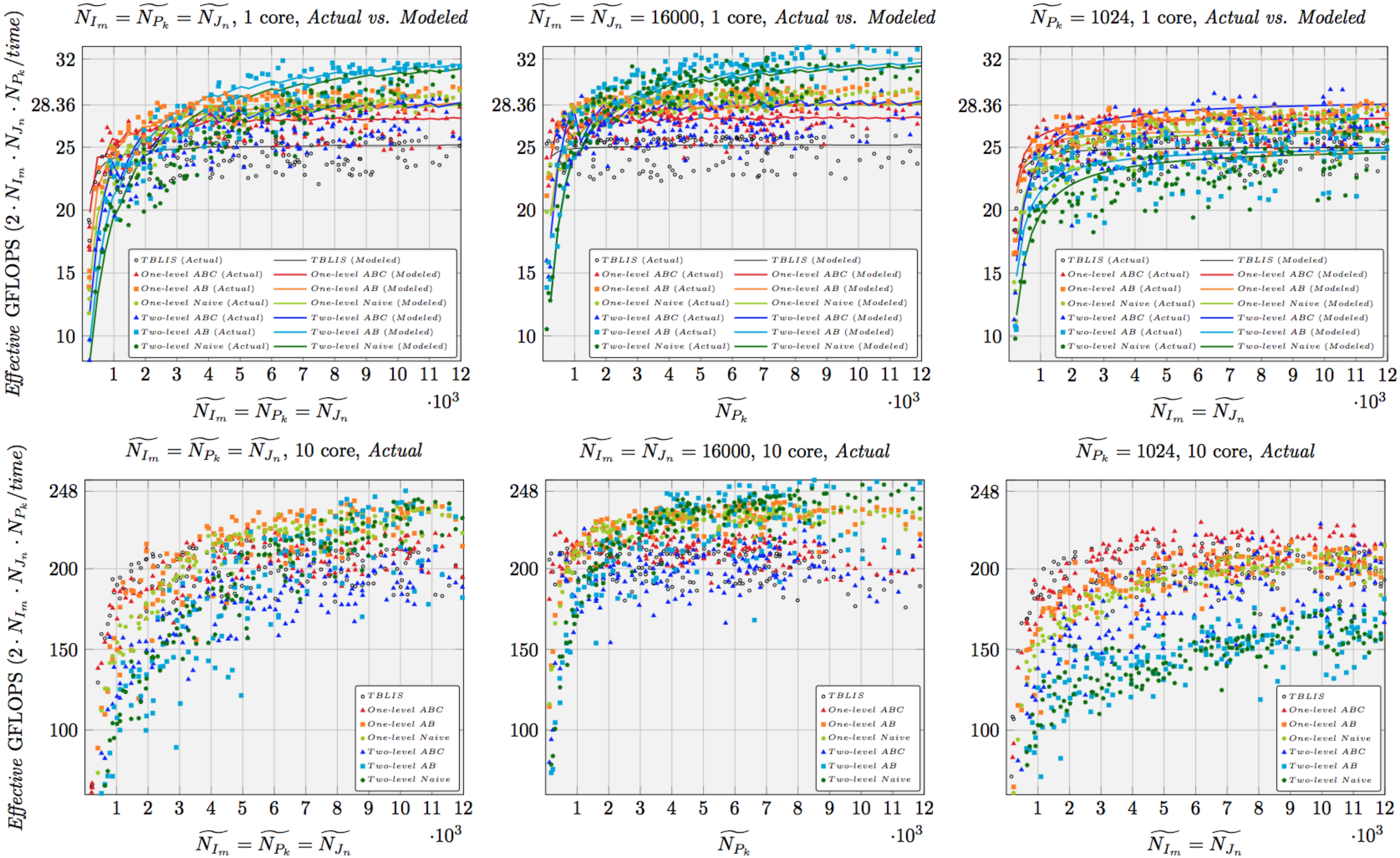
$$N_{I_m} = \prod_{i \in I_m} N_i = N_{i_0} \cdot \ldots \cdot N_{i_{m-1}},$$
$$N_{J_n} = \prod_{j \in J_n} N_j = N_{j_0} \cdot \ldots \cdot N_{j_{n-1}},$$
$$N_{P_k} = \prod_{p \in P_k} N_p = N_{p_0} \cdot \ldots \cdot N_{p_{k-1}}.$$





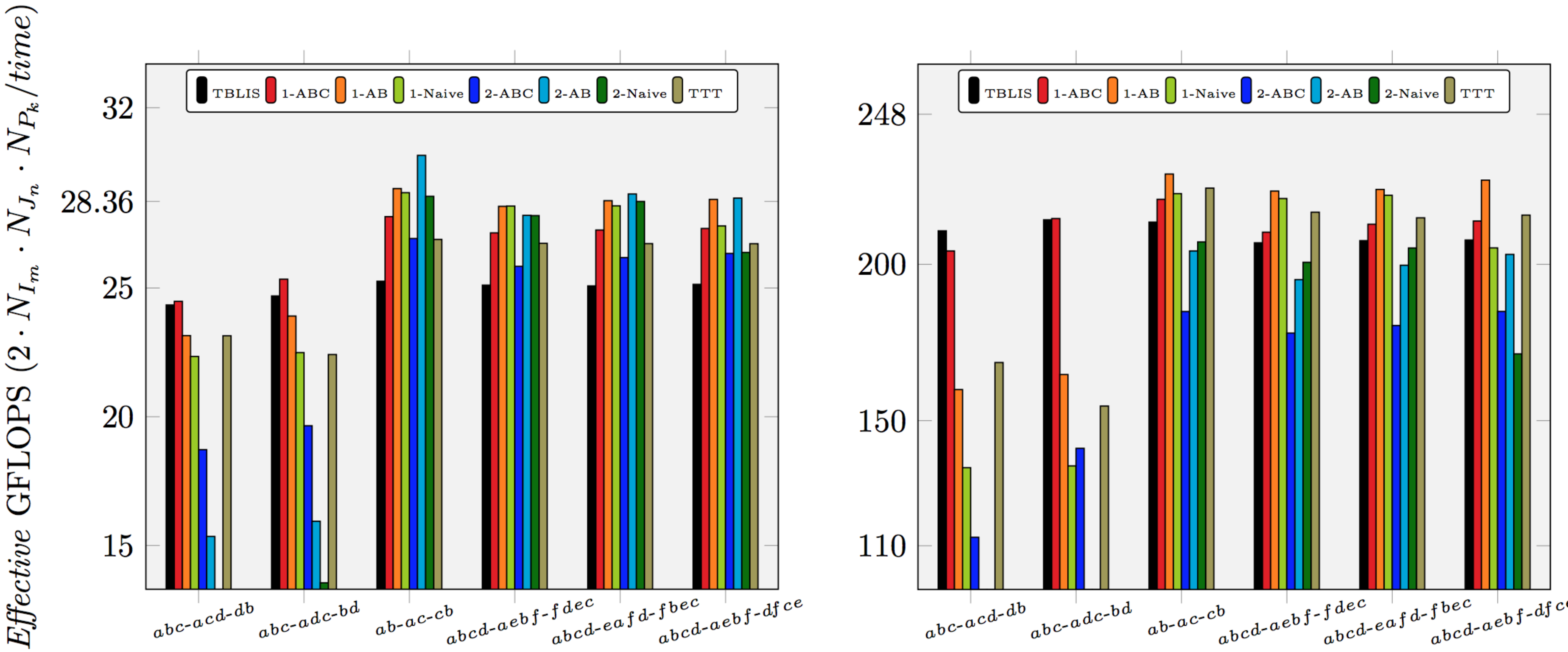
Synthetic Tensor Contractions

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



Real-world Benchmark

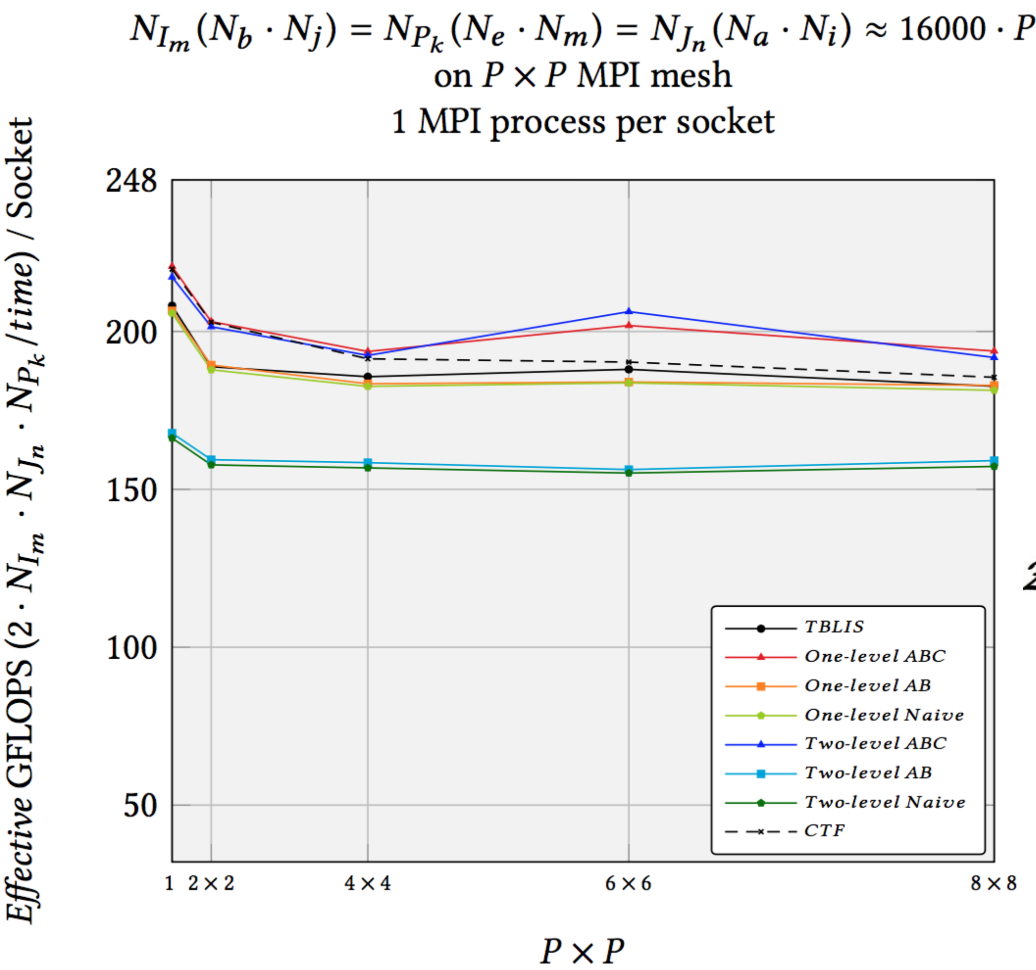
Intel Xeon E5-2680 v2 (Ivy Bridge, 10 core/socket)



$\mathcal{C}_{abcd} \mathrel{+}= \mathcal{A}_{aebf} \mathcal{B}_{dfce}$ is denoted as $abcd\text{-}aebf\text{-}dfce$

Paul Springer, and Paolo Bientinesi. "Design of a high-performance GEMM-like tensor-tensor multiplication." arXiv preprint arXiv:1607.00145 (2016).

Distributed Memory Experiments



Weak scalability performance result of the various implementations for a 4-D tensor contraction CCSD application on distributed memory:

$$\mathcal{Z}_{abij} \mathrel{+}= \mathcal{W}_{bmej} \mathcal{T}_{aeim}$$

$$\mathcal{Z} \mathrel{+}= \left(\mathcal{W}_{e;0} \mid \cdots \mid \mathcal{W}_{e;K-1} \right) \begin{pmatrix} \mathcal{T}_{e;0} \\ \hline \vdots \\ \hline \mathcal{T}_{e;K-1} \end{pmatrix}$$

CTF shows the performance of the Cyclops Tensor Framework (linked with Intel MKL).

Solomonik, Edgar, Devin Matthews, Jeff Hammond, and James Demmel. "Cyclops tensor framework: Reducing communication and eliminating load imbalance in massively parallel contractions." In Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, pp. 813-824. IEEE, 2013.

Outline

- Background
  - High-performance GEMM
  - High-performance Strassen
  - High-performance Tensor Contraction
- Strassen's Algorithm for Tensor Contraction
- Performance Model
- Experiments
- Conclusion

Conclusion

- First work to leverage Strassen’s Algorithm for Tensor Contraction.
- Fusing matrix summation and permutations with packing and micro-kernel operations inside GEMM.
- Avoiding explicit transpositions and extra workspace, and reducing the overhead of memory movement.
- Achieving ~1.3x speedup on single core, multicore, and distributed memory parallel architectures.

Acknowledgement

- NSF grants ACI-1550493, CCF-1714091.
- 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 Martin Schatz for his help with distributed memory implementations, 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/tblis-strassen