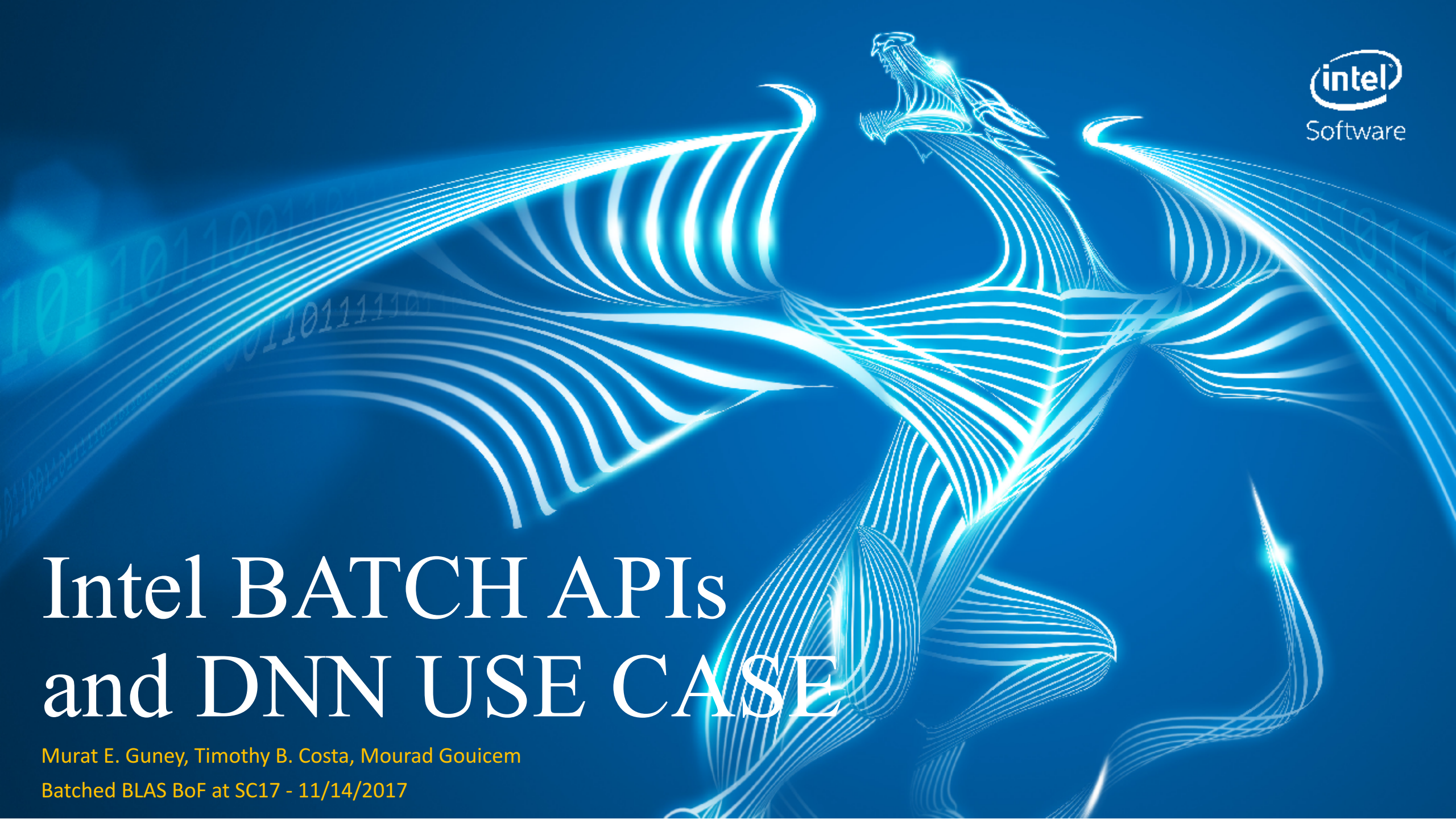


Intel BATCH APIs and DNN USE CASE

Murat E. Guney, Timothy B. Costa, Mourad Gouicem

Batched BLAS BoF at SC17 - 11/14/2017

Intel MKL Solutions for SMALL, MEDIUM, and SKEWED Sizes

- MKL_DIRECT_CALL
  - Improves performance for small sizes (M, N, K < 20)
  - Skips error checking and function call overheads
  - Enabled for several functions
    - BLAS: gemm, gemm3m, syrk, trsm, axpy, dot
    - LAPACK: potrf, getrf, getrs, getri, geqrf
- Batch APIs
  - Improves performance for small-medium sizes (M, N, K < 500)
  - Groups several independent function calls together
  - Enabled for gemm, gemm3m and trsm BLAS functions
- Packed APIs
  - Improves performance for small-medium M or N sizes (M or N < 500)
  - Allows amortizing copy overheads over several GEMM calls with same input matrix
  - Enabled for sgemm and dgemm BLAS functions
- Compact APIs
  - Improves performance for small sizes (M, N, K < 20)
  - Enables vectorization over very small matrix dimensions by reformatting the data in a compact layout
  - Enabled for several functions
    - BLAS: gemm, trsm
    - LAPACK: getrinp, getrfnp, potrf, geqrf

Batch API

- Execute independent BLAS operations simultaneously with one function call
- Ensure no data dependency between the operations
- Take advantage of all cores even for small-medium sizes (M, N, K < 500)
- Minimize library overheads
- Some code modification is required to group same size matrices together

$$\mathbf{C}^1 = alpha \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^1) + beta \cdot \mathbf{C}^1$$

$$\mathbf{C}^2 = alpha \cdot \mathrm{op}(\mathbf{A}^2) \cdot \mathrm{op}(\mathbf{B}^2) + beta \cdot \mathbf{C}^2$$

$$\mathbf{C}^3 = alpha \cdot \mathrm{op}(\mathbf{A}^3) \cdot \mathrm{op}(\mathbf{B}^3) + beta \cdot \mathbf{C}^3$$

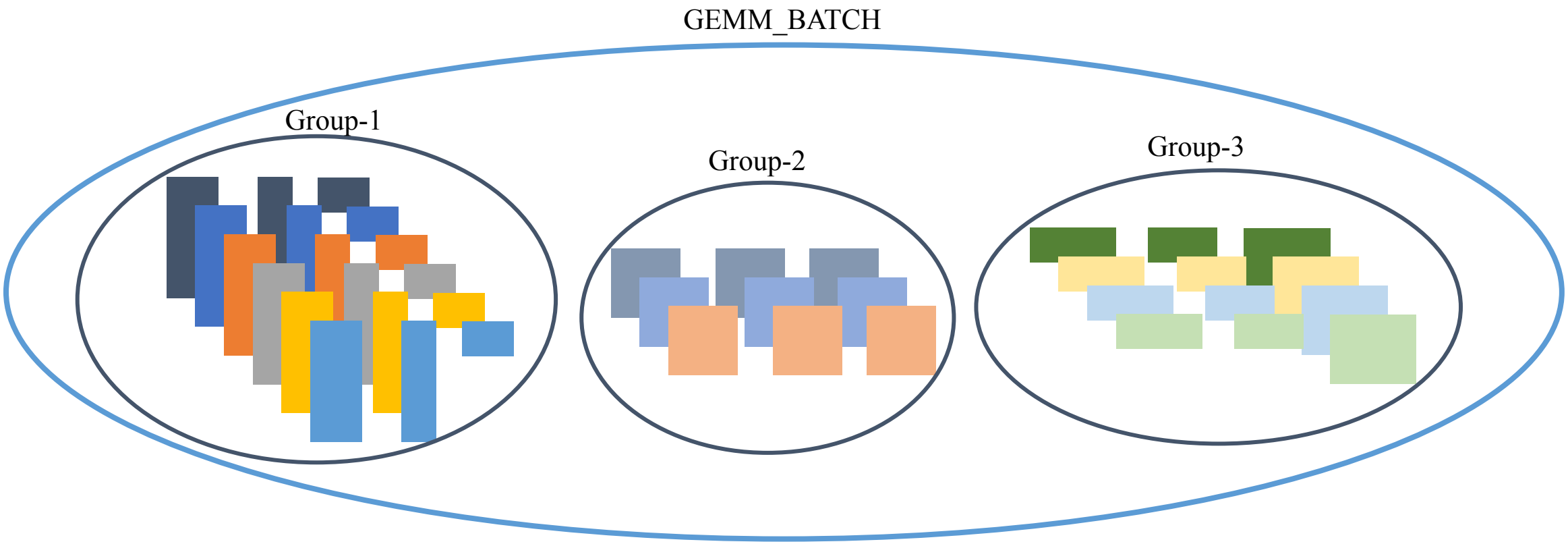
Execute in parallel assuming no pointer aliasing

$$\mathbf{C}^2 = alpha \cdot \mathrm{op}(\mathbf{A}^4) \cdot \mathrm{op}(\mathbf{B}^4) + beta \cdot \mathbf{C}^2$$

Wait for a previous write to $\mathbf{C}^2$

Group Concept in Batch API

- Group: a set of GEMM operations with same input parameters (matrix pointers can be different)
  - Transpose, size, leading dimensions, alpha and beta
- One GEMM_BATCH call can handle one or more groups

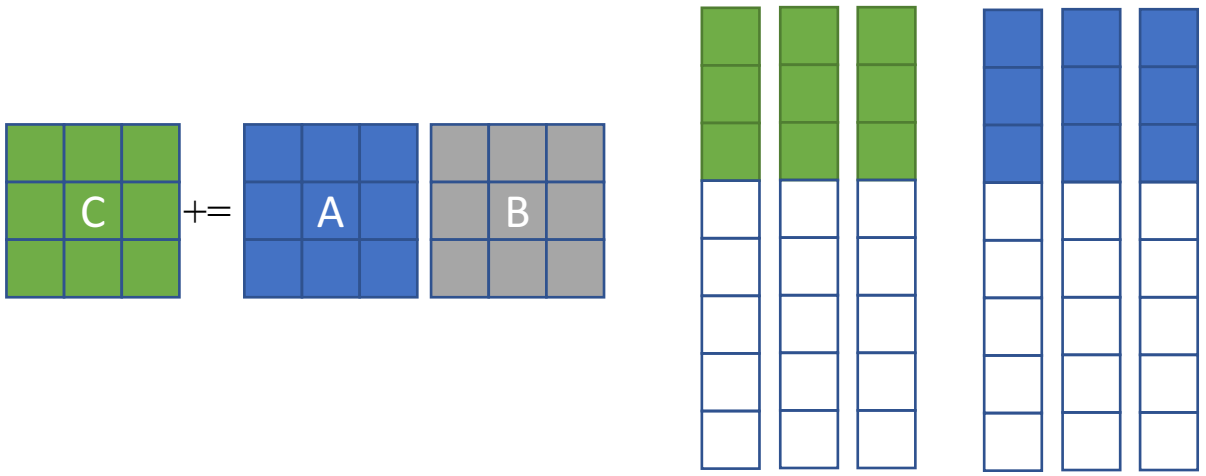


VECTORIZATION CHALLENGES with Small MATRICES

- Limited vectorization opportunity and non-local data access for large leading dimensions

```
C = beta*C
DO i=1,(M/u)
   DO j=1,N
      DO kk=1,K
         C(i  ,j) += alpha*A(i,kk)*B(kk,j)
         C(i+1,j) += alpha*A(i+1,kk)*B(kk,j)
           .
           .
           .
         C(i+u,j) += alpha*A(i+u,kk)*B(kk,j)
      END DO
   END DO
END DO
```

3x3x3 DGEMM and Intel® AVX512 register mapping

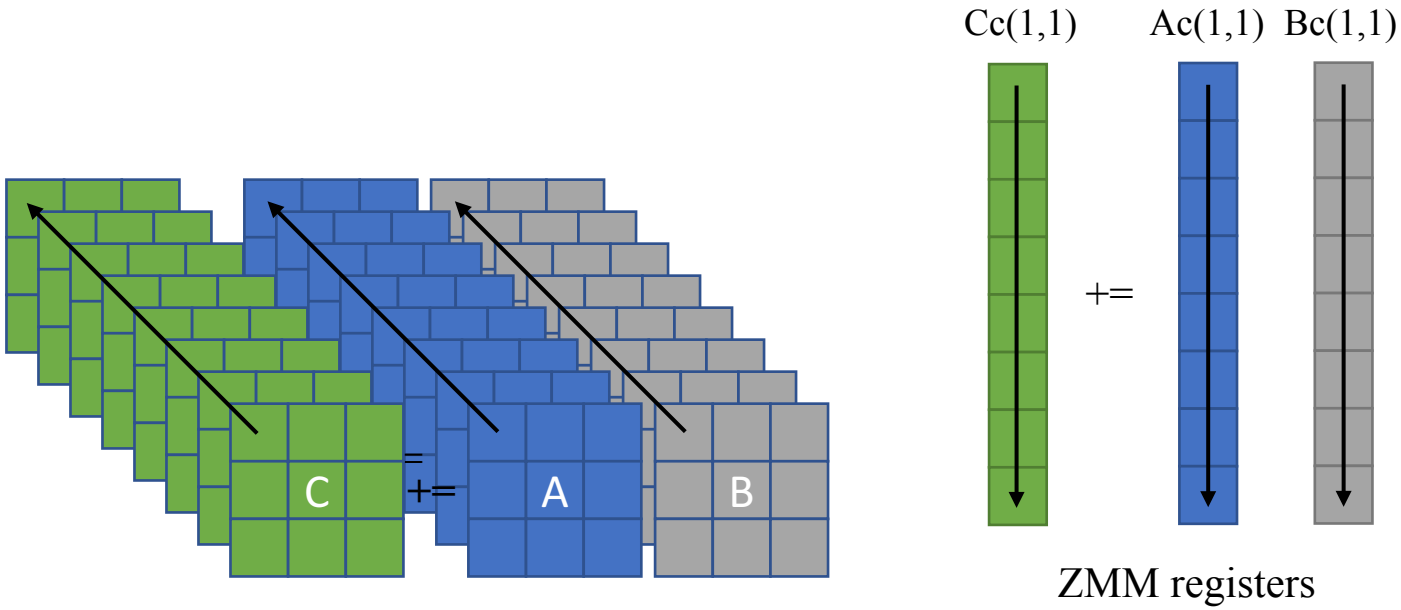


ZMM registers

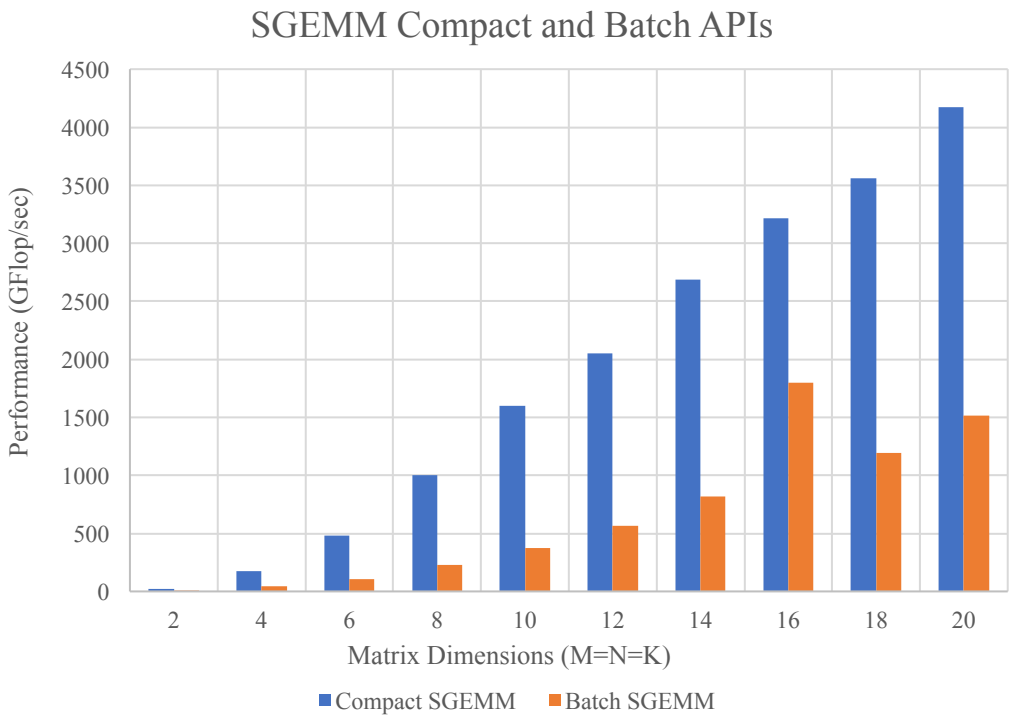
VECTORIZATION WITH COMPACT DATA LAYOUT

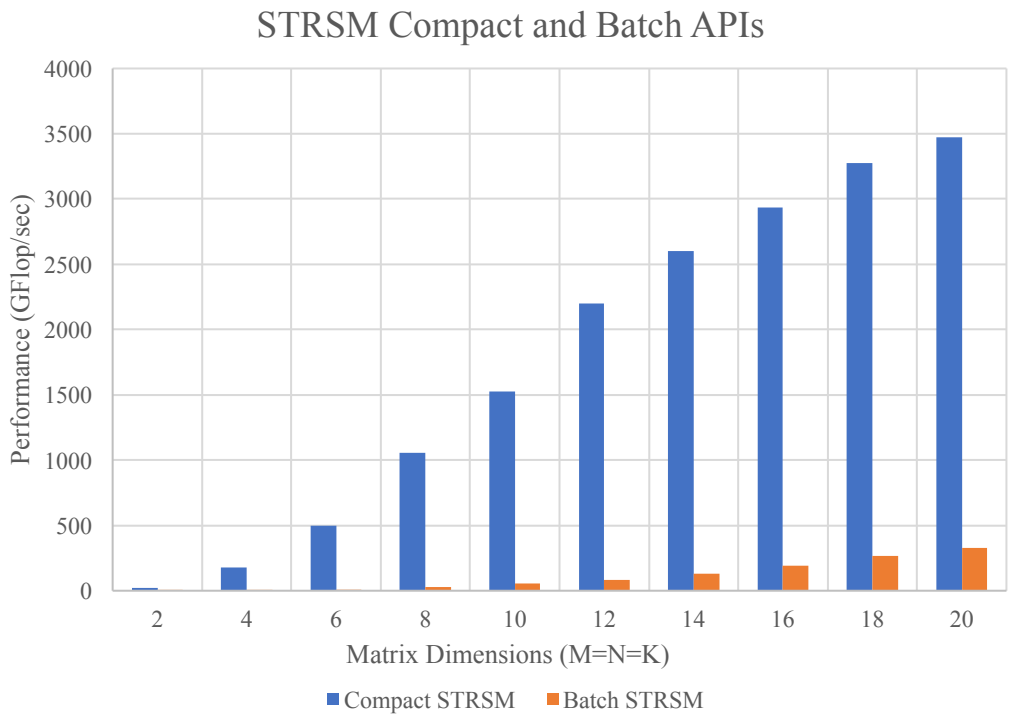
- Matrix elements with same col/row index loaded to a SIMD register
- Vectorization across the matrices becomes trivial
- Data padding if the number of matrices are not multiples of SIMD vector length

3x3x3 MKL_DGEMM_COMPACT and Intel® AVX512 register mapping



COMPACT API Performance On Intel® Xeon® Platinum Processor





Configuration: Intel® Xeon® Platinum 8180, 2x28 cores, 2.5 GHz, 376 GB RAM, OS Ubuntu, 16.04 LTS; Intel® MKL 2018. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Benchmark source: Intel Corporation. **Optimization Notice:** Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804

Packed API

- GEMM may copy the input matrices into internal buffers for efficient computation
- Copy operation is costly for medium or skewed sizes (M or N < 500)
- Amortize copy (pack) operation over multiple GEMM calls with the same input matrix
- Copy (pack) the data once and reuse it in many GEMM calls
- Improves the performance when there is input matrix reuse

$$\mathbf{C}^1 = alpha \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^1) + beta \cdot \mathbf{C}^1$$

$$\mathbf{C}^2 = alpha \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^2) + beta \cdot \mathbf{C}^2$$

$$\mathbf{C}^3 = alpha \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^3) + beta \cdot \mathbf{C}^3$$

Input matrix A^1 is shared between three GEMM calls

USE CASE for Batched Pack API

- Winograd-based convolution:

  Parameters: a(4 or 6), b([32,512]), ntiles ([4, 10k]), ic, oc ([64,512])
  Input: 4DTensor Inp1, Inp2
  Output: 4DTensor Out

  1. As[a][a][m][k] = transform1(Inp1)
  2. Bs[a][a][n][k] = transform2(Inp2)
  3. For e=0 to a
     For f=0 to a
     For g=0 to b
     Cs[e][f][g] = GEMM(As[e][f][g], Bs[e][f])
  4. Out = dst_transform(Cs)

GEMM Sizes:
Forward: m=oc, n=ntiles, k=ic,
Bwd data: m=ic, n=ntiles, k=oc,
Bwd weight: m=oc, n=ic, k=ntiles

- Performance opportunities: B matrix can be packed during data transform
- Distributed memory libraries (SLATE and ScaLAPACK) also requested for packed batch

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel
Software