Software

ADVENTURES IN BATCHED LINEAR ALGEBRA IN INTEL® MATH KERNEL LIBRARY

Sarah Knepper, Peter Caday, Kazushige Goto, Louise Huot,
Mesut Meterelliyoz, Arthur Mitrano, and Shane Story, Intel® MKL

Agenda

- Performance challenges and solutions
- Batched linear algebra
- Alternative data layout: compact APIs
- Just-in-time (JIT) generation
- Summary and Intel MKL resources

How Intel MKL gets performance

More Cores → More Threads → Wider Vectors

	Intel® Xeon® Processor 64-bit	Intel® Xeon® Processor 5100 series	Intel® Xeon® Processor 5500 series	Intel® Xeon® Processor 5600 series	Intel® Xeon® Processor E5-2600 v2 series	Intel® Xeon® Processor E5-2600 v3 series v4 series	Intel® Xeon® Scalable Processor[1]	Intel® Xeon Phi™ x200 Processor (KNL)
Up to Core(s)	1	2	4	6	12	18-22	28	72
Up to Threads	2	2	8	12	24	36-44	56	288
SIMD Width	128	128	128	128	256	256	512	512
Vector ISA	Intel® SSE3	Intel® SSE3	Intel® SSE4-4.1	Intel® SSE 4.2	Intel® AVX	Intel® AVX2	Intel® AVX-512	Intel® AVX-512

1. Product specification for launched and shipped products available on ark.intel.com.

Classification of sizes and performance challenges

Small Sizes

- M, N, K < 20
- Challenges: High function call overheads, low vectorization, low parallelization
- Solutions: Batch API, Compact API, MKL_DIRECT_CALL, and JIT GEMM

Medium Sizes

- 20 < M, N, K < 500
- Challenges: Low parallelization, high copy overheads
- Solutions: Batch API and Pack API

Large Sizes

- M, N, K > 5000
- Performance close to machine's theoretical peak

Introduction to batched linear algebra

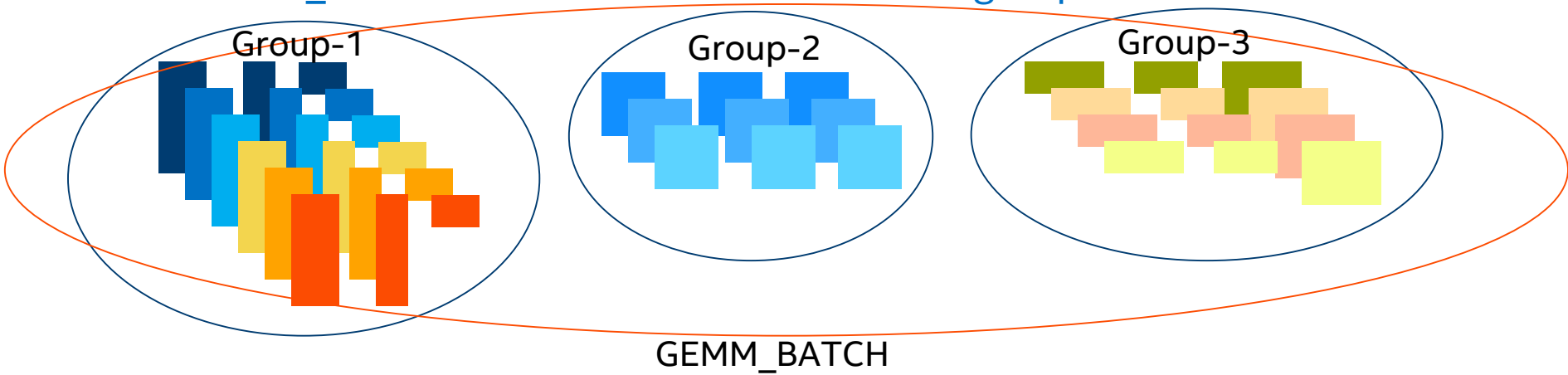
- Execute independent general matrix multiplication (GEMM) operations simultaneously with one function call
- Ensure no data dependency between the operations
- Take advantage of all cores even for small-medium sizes (M, N, K < 500)
- Minimize library overheads
- Some code modification is required to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

$$\left.\begin{aligned} \mathbf{C}^1 &= \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^1) + \mathit{beta} \cdot \mathbf{C}^1 \\ \mathbf{C}^2 &= \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^2) \cdot \mathrm{op}(\mathbf{B}^2) + \mathit{beta} \cdot \mathbf{C}^2 \\ \mathbf{C}^3 &= \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^3) \cdot \mathrm{op}(\mathbf{B}^3) + \mathit{beta} \cdot \mathbf{C}^3 \end{aligned}\right\} \text{Execute in parallel assuming no pointer aliasing}$$

$$\left.\mathbf{C}^2 = \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^4) \cdot \mathrm{op}(\mathbf{B}^4) + \mathit{beta} \cdot \mathbf{C}^2\right\} \text{Wait for a previous write to } \mathbf{C}^2$$

Group concept in batch API

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



API advantages of groups

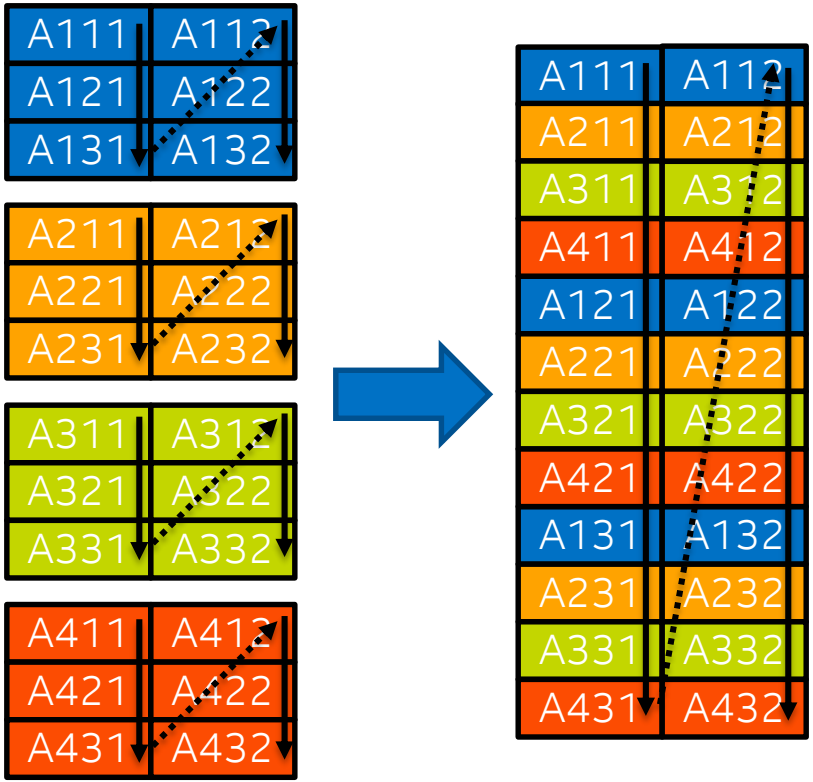
- Same function can handle 1 or more groups
- Consistent level of redirection for function parameters
  - Integer → array of integers
  - Pointer → array of pointers
- Smaller parameter arrays
- Minimal error checking needed

Performance opportunities for batching with groups

- Minimize library overheads for small sizes
  - Reduce function call overheads
  - Do error checking once for each GEMM group
- Better exploit parallelism available in many-/multi-core processors
- Potential performance techniques
  - Prefetch the matrix blocks across GEMM calls
  - Pack matrix once if used throughout group
  - Just-in-time (JIT) generation to create GEMM kernel for group

Compact API

- A new data layout to better utilize vectorization on large numbers of same size matrices
- Matrix elements with same index are interleaved in memory
- Size of the subgroup is SIMD length to fully utilize SIMD instructions
- Example reformatting of 3x2 matrices with subgroup size = 4:



Compact APIs Workflow

Transform the data from standard format into compact format:

```
MKL_COMPACT_PACK compact_format = mkl_get_format_compact();

a_size = mkl_dget_size_compact(lda, k, compact_format, num_matrix);

mkl_dgepack_compact(layout, m, k, a_array, lda, a_c, lda, compact_format,
                    num_matrix);
```

Perform multiple operations on the compact data:

```
mkl_dgemm_compact(layout, transa, transb, m, n, k, alpha, a_c, lda, b_c, ldb,
                  beta, c_c, ldc, compact_format, num_matrix);
```

Transform the data from compact format into standard format:

```
mkl_dgeunpack_compact(layout, m, n, c_array, ldc, c_c, ldc, compact_format,
                      num_matrix);
```

JIT (Just-In-Time) GEMM Overview

- Generate (JIT) a customized kernel at run time to decrease overheads
- Use preprocessor macro MKL_DIRECT_CALL_JIT or MKL_DIRECT_CALL_SEQ_JIT
  - No changes to user code
  - Intel MKL may JIT a specific kernel
  - Kernels are stored in an internal hash table to amortize cost of generation
- Use new JIT APIs
  - User responsible for managing kernels
  - Eliminates more overheads for even better performance

JIT APIs Workflow

Create a handle and generate GEMM kernel:

```
mkl_jit_status_t status = mkl_jit_create_sgemm(&jit_handle,
    layout, transa, transb, m, n, k, alpha, lda, ldb, beta, ldc);
```

Get kernel associated with handle:

```
sgemm_jit_kernel_t kernel = mkl_jit_get_sgemm_ptr(jit_handle);
```

Repeatedly execute the GEMM kernel:

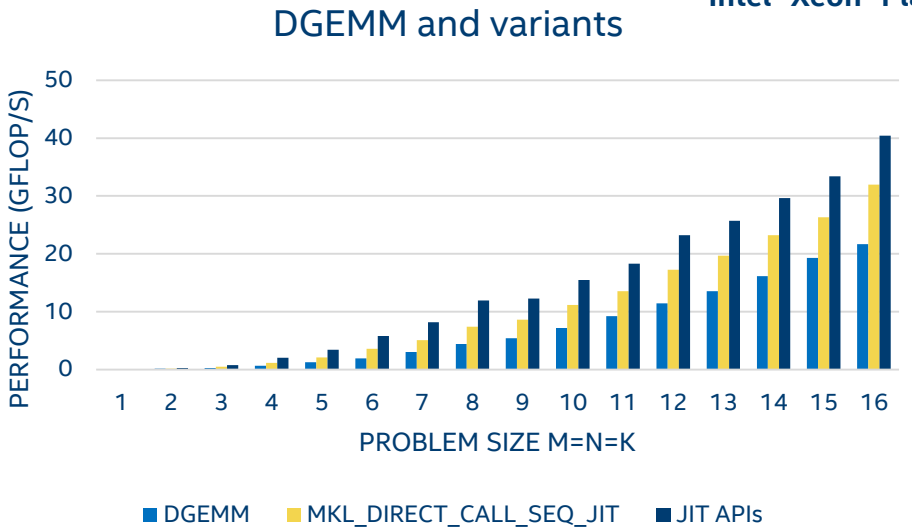
```
kernel(jit_handle, a, b, c);
```

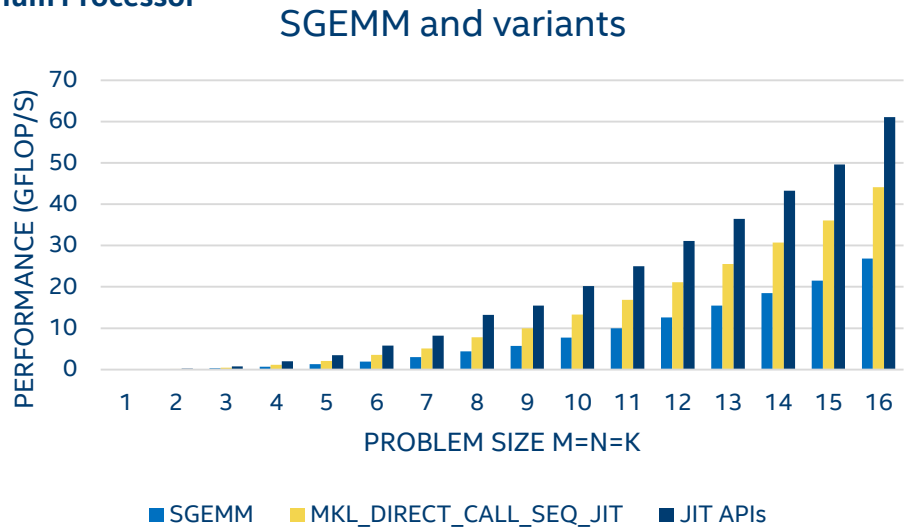
Destroy the created handle/GEMM kernel:

```
mkl_jit_destroy(jit_handle);
```

JIT DGEMM, SGEMM on Intel® Xeon® Platinum Processor

Optimized by Intel® Math Kernel Library 2019 Update 2 for Intel® Xeon® Platinum Processor





Performance results are based on testing as of February 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.
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.
Configuration: Testing by Intel as of February 2019: Intel® Xeon® Platinum 8180 H0 205W 2x28@2.5GHz 192GB DDR4-2666
Benchmark Source: Intel® Corporation.
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.

Final Remarks

- Small/medium matrices have various performance challenges
- Batching better utilizes multi- and many-cores for small/medium matrices
- Groups contain matrices with same parameters (size, leading dimension, etc.)
- Intel MKL batch API combines ease-of-use with performance opportunities
- Can interleave data to increase vectorization across matrices
- Intel MKL compact API can significantly improve performance for small sizes
- Just-in-time (JIT) generation can reduce overheads for small sizes, by creating custom kernel at run-time and re-using it many times

Intel MKL resources

- Developer Reference: https://software.intel.com/en-us/articles/mkl-reference-manual
- Release Notes: https://software.intel.com/en-us/articles/intel-math-kernel-library-release-notes-and-new-features
- Forum: https://software.intel.com/en-us/forums/intel-math-kernel-library
- No cost option for Intel MKL: https://software.intel.com/en-us/articles/free-mkl
- Intel MKL for Deep Neural Networks (Intel MKL-DNN): https://github.com/intel/MKL-DNN

Legal Disclaimer & Optimization Notice

Performance results are based on testing as of February 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

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.

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.

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