Compact Batched BLAS

Intel® MKL Team - February 25, 2017

Outline

- Intel® Math Kernel Library (Intel® MKL) Batched BLAS
- Compact Batched BLAS
  - Limitations of Batched BLAS for very small matrices
  - Compact format: an alternative data layout for small sizes
  - Compact Batched API
    - Compact matrix struct
    - Data manipulation
    - Compact BLAS/LAPACK function APIs
  - Performance

Intel MKL Batched BLAS

Overview of Intel MKL Batched BLAS API

The API allows batching BLAS operations with different parameters

- Group: a number of BLAS operations with same parameters
- Batch: a number of BLAS groups
- <function>_BATCH executes multiple groups simultaneously

Two additional parameters to the traditional GEMM functions

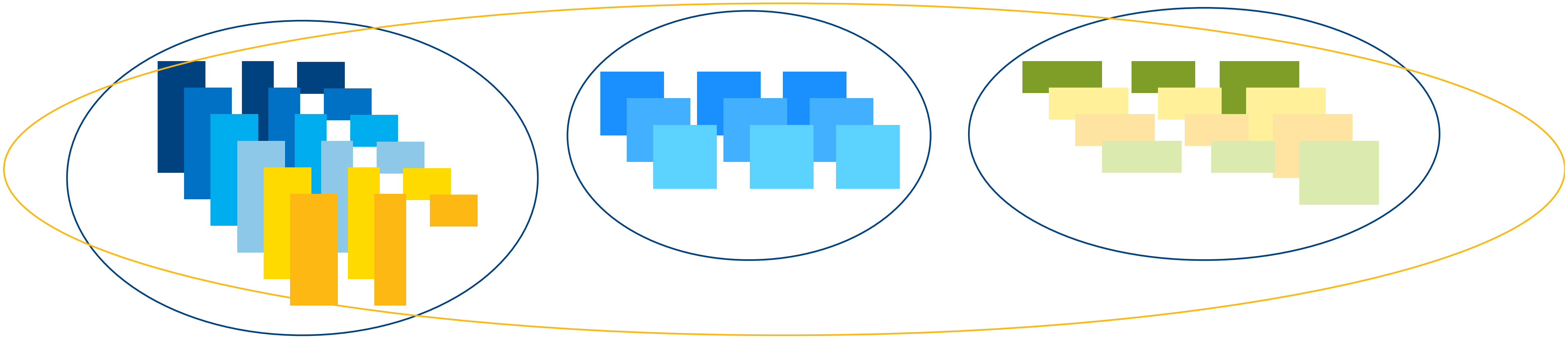
- group_count (integer): total number of groups
- group_size (integer array): the number of GEMMs within each group

A consistent level of redirection for GEMM parameters

- integer becomes *array* of integers
- Matrix pointer becomes *array* of matrix pointers

Intel MKL Group Concept

- Group: set of BLAS operations with same input parameters (except for matrix pointers)
  - Transpose, size, leading dimension, alpha, beta
- One or more groups per <function>_BATCH call



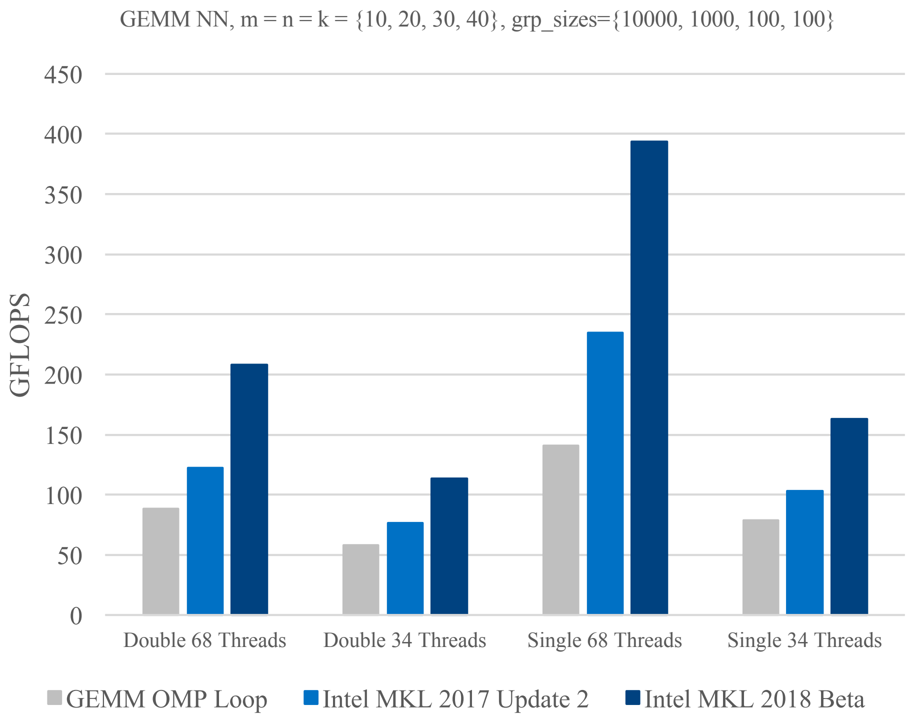
Comparison of various batched GEMMs

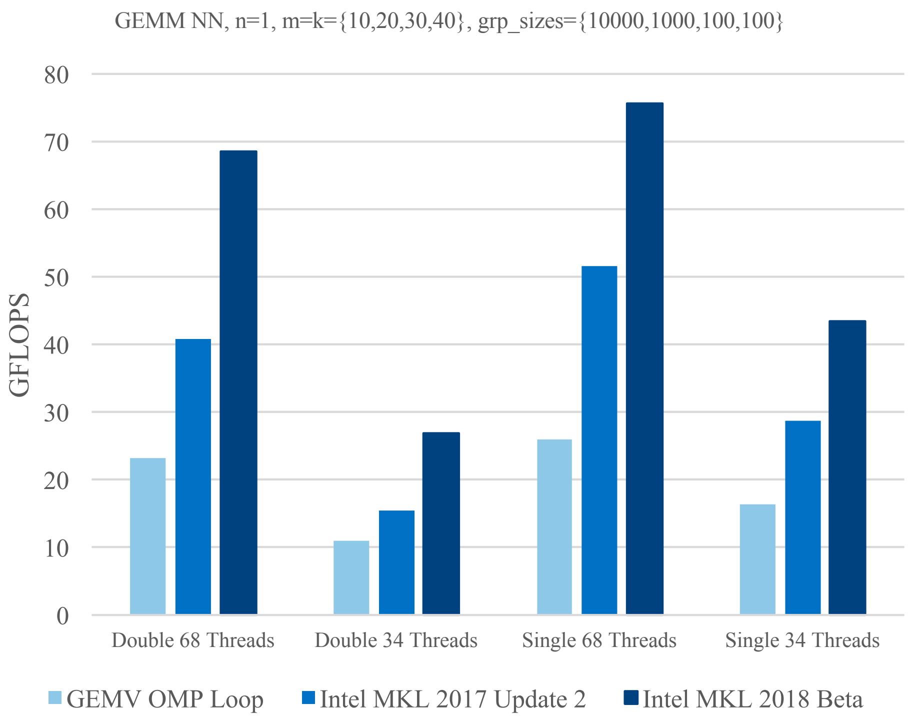
Argument	Description	BLAS sgemm	magma_sgemm_batched	NVidia cublasSgemmBatched	UTK sgemm_batch	Intel MKL sgemm_batch
HANDLE	handle to the cuBLAS library context	--	--	cublasHandle_t	--	--
TRANSA	op(A)	char	char	char	char *	char *
TRANSB	op(B)	char	char	char	char *	char *
M	rows of op(A)/C	int	int	int	int *	int *
N	columns of op(B)/C	int	int	int	int *	int *
K	columns of op(A)/rows of op(B)	int	int	int	int *	int *
ALPHA	alpha	float	float	float *	float *	float *
A	input matrix	float *	float **	float **	float **	float **
LDA	leading dimension of A	int	int	int	int *	int *
B	input matrix	float *	float **	float **	float **	float **
LDB	leading dimension of B	int	int	int	int *	int *
BETA	beta	int	float	float *	float *	float *
C	input/output matrix	float *	float **	float **	float **	float **
LDC	leading dimension of C	int	int	int	int *	int *
BATCHCOUNT	number of matrices	--	int	int	int	--
QUEUE	queue to execute in	--	magma_queue_t	--	--	--
BATCH_OPTS	style for batched (fixed or variable)	--	--	--	enum	--
INFO	error handling	--	--	--	int *	--
GROUP_COUNT	number of groups	--	--	--	--	int
GROUP_SIZES	number of matrices in each group	--	--	--	--	int *

For simplicity, some enum types reduced to char or int. Table idea and some data from Performance, Design, and Autotuning of Batched GEMM for GPUs by Ahmad Abdelfattah, Azzam Haidar, Stanimire Tomov, and Jack Dongarra.

Performance Improvements

- Intel MKL 2018 Beta
  - Performance improved for ?GEMM_BATCH on all architectures.
  - Greatly improved performance for N==1 ?GEMM_BATCH.





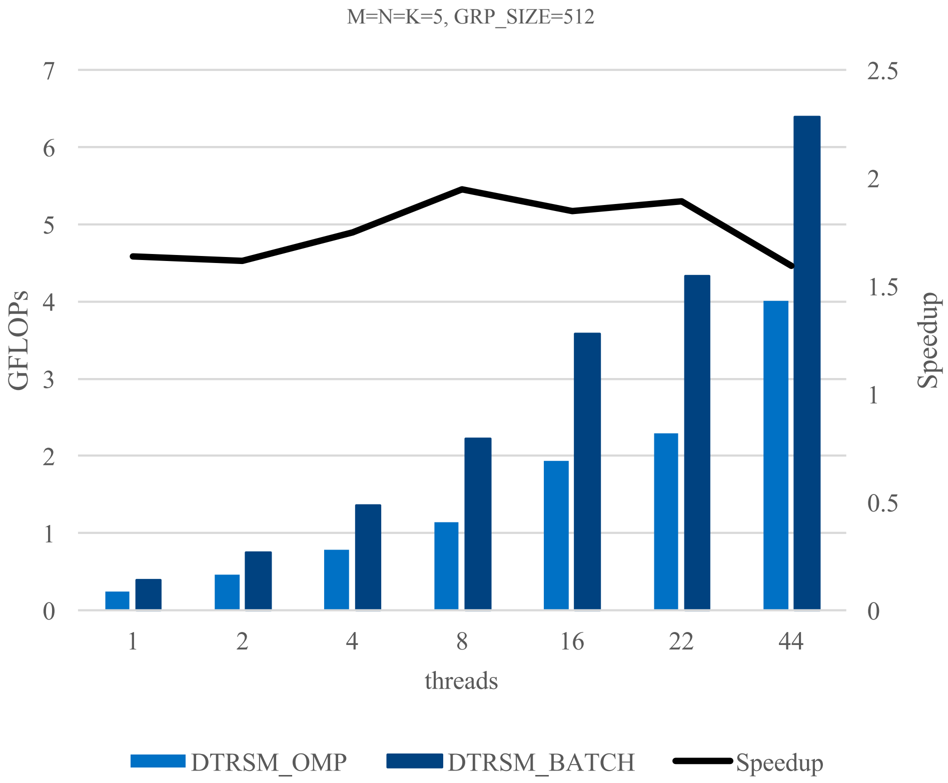
Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 2018 Beta, Intel® MKL 2017 Update 2; Hardware: Intel® Xeon Phi™ Processor 7250, 68 cores (34 MB total cache, 1.4GHz), 16GB MCDRAM Memory, 96GB of DDR4 Memory; Operating System: RHEL 7.2 GA x86_64

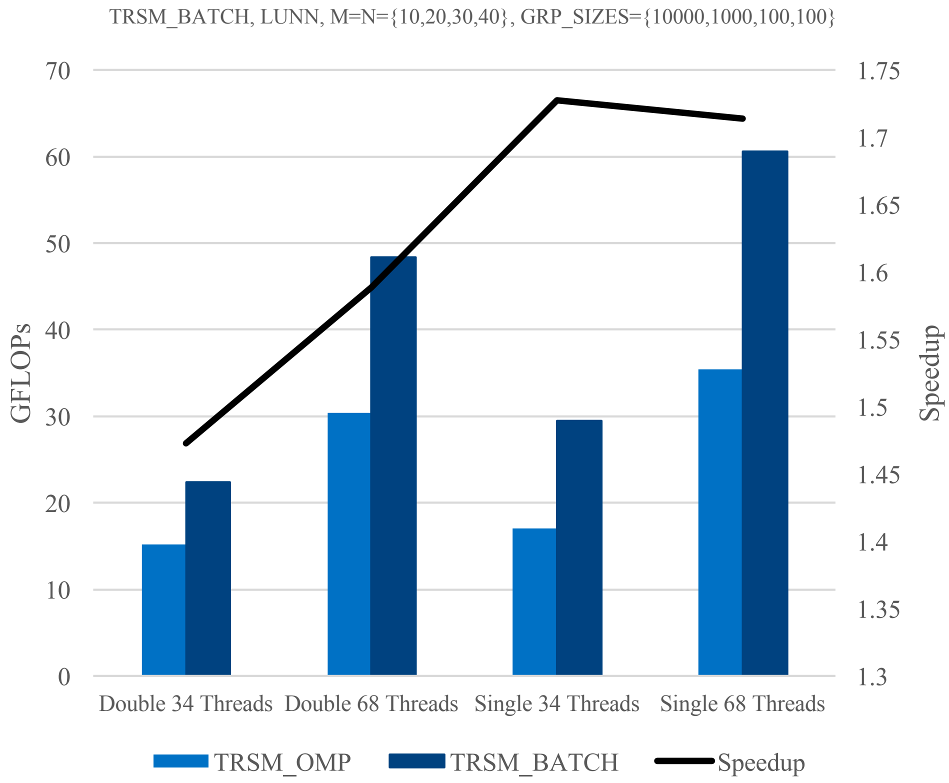
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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. * Other brands and names are the property of their respective owners. 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 .

New Feature: batched TRSM

- Intel MKL 2018 Beta includes ?TRSM_BATCH





Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 2018 Beta, ; Hardware: Intel® Xeon™ Processor E5-2699 v4, 2 22-core CPUs (55 MB cache, 2.2 GHz), 64GB of DDR4 Memory; Operating System: RHEL 7 GA x86_64

Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 2018 Beta; Hardware: Intel® Xeon Phi™ Processor 7250, 68 cores (34 MB total cache, 1.4GHz), 16GB MCDRAM Memory, 96GB of DDR4 Memory; Operating System: RHEL 7.2 GA x86_64

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. * Other brands and names are the property of their respective owners. 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 .

Benefits & limitations of batched BLAS

For medium and small sizes:

- Schedule simultaneous BLAS functions on Intel® Xeon® and Intel® Xeon Phi™
  - Assign optimal number of threads/cores to each operation

For small sizes:

- Limit function call and error checking overhead for small sizes
  - Check for error and dispatch once, run kernels many times

Limitation:

- HPC applications often operate on large numbers of very small matrices (3x3, 5x5, 6x6, 9x9, 15x15)
  - e.g. FEM models, preconditioner application, computational lithography, collaborative filtering
- Limited benefit from vectorization in kernels

Solution:

- Potential for large gains from non-standard data layouts
- Cross-matrix vectorization

Compact Batched BLAS/LAPACK

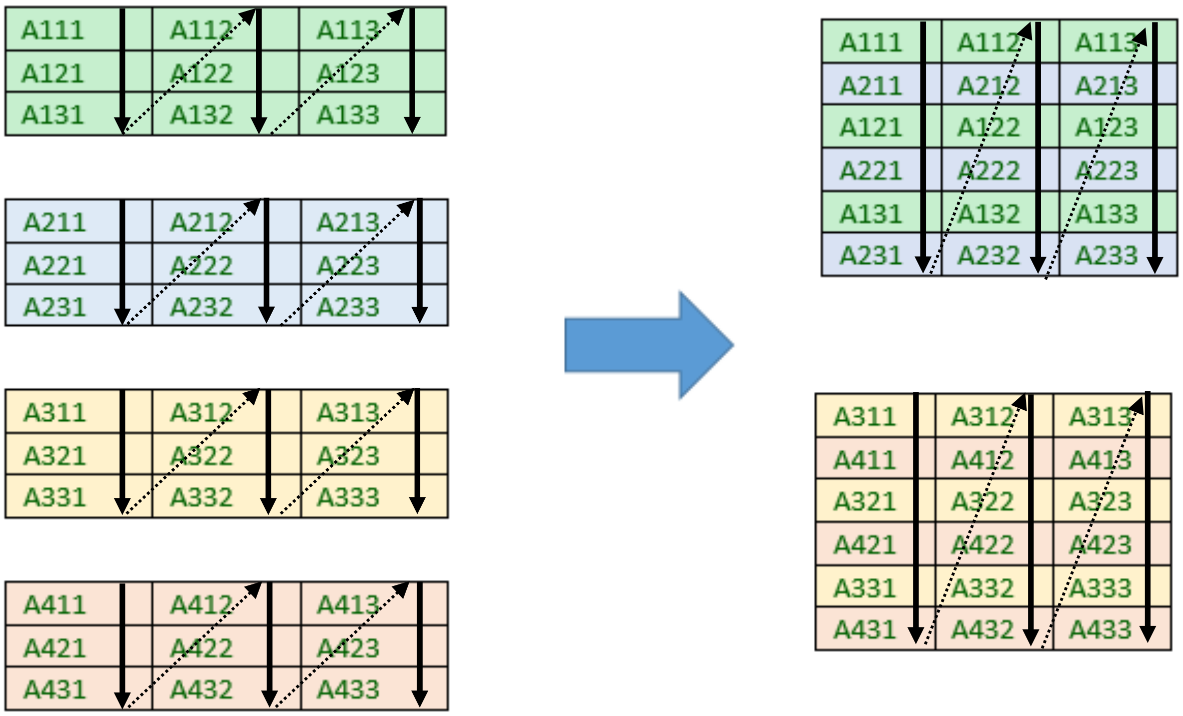
Compact Batched BLAS/LAPACK API overview

- Compact: “Closely and neatly packed together, dense.”
- Compact Batched BLAS API:
  - Matrix subgroups are weaved together for cross-matrix vectorization
  - Designed for performance for small sizes
  - Up to 11x over existing MKL batched BLAS in early testing
- Two use cases:
  - Applications with data already in compact format call compact batched compute functions directly for any batched operations.
  - Applications with traditional data layout that will perform several BLAS operations on a batch of matrices first call MKL provided pack functions to set up data. The data manipulation cost is amortized by re-use of matrices.

- Acknowledgement: the Compact API was motivated by discussions with the KokkosKernels team at Sandia National Laboratory.

Compact Data layout details

- Consistent with KokkosKernels and other community formatting
- Consistent layout for all BLAS/LAPACK routines / matrices.



A111	A112	A113
A121	A122	A123
A131	A132	A133

A211	A212	A213
A221	A222	A223
A231	A232	A233

A311	A312	A313
A321	A322	A323
A331	A332	A333

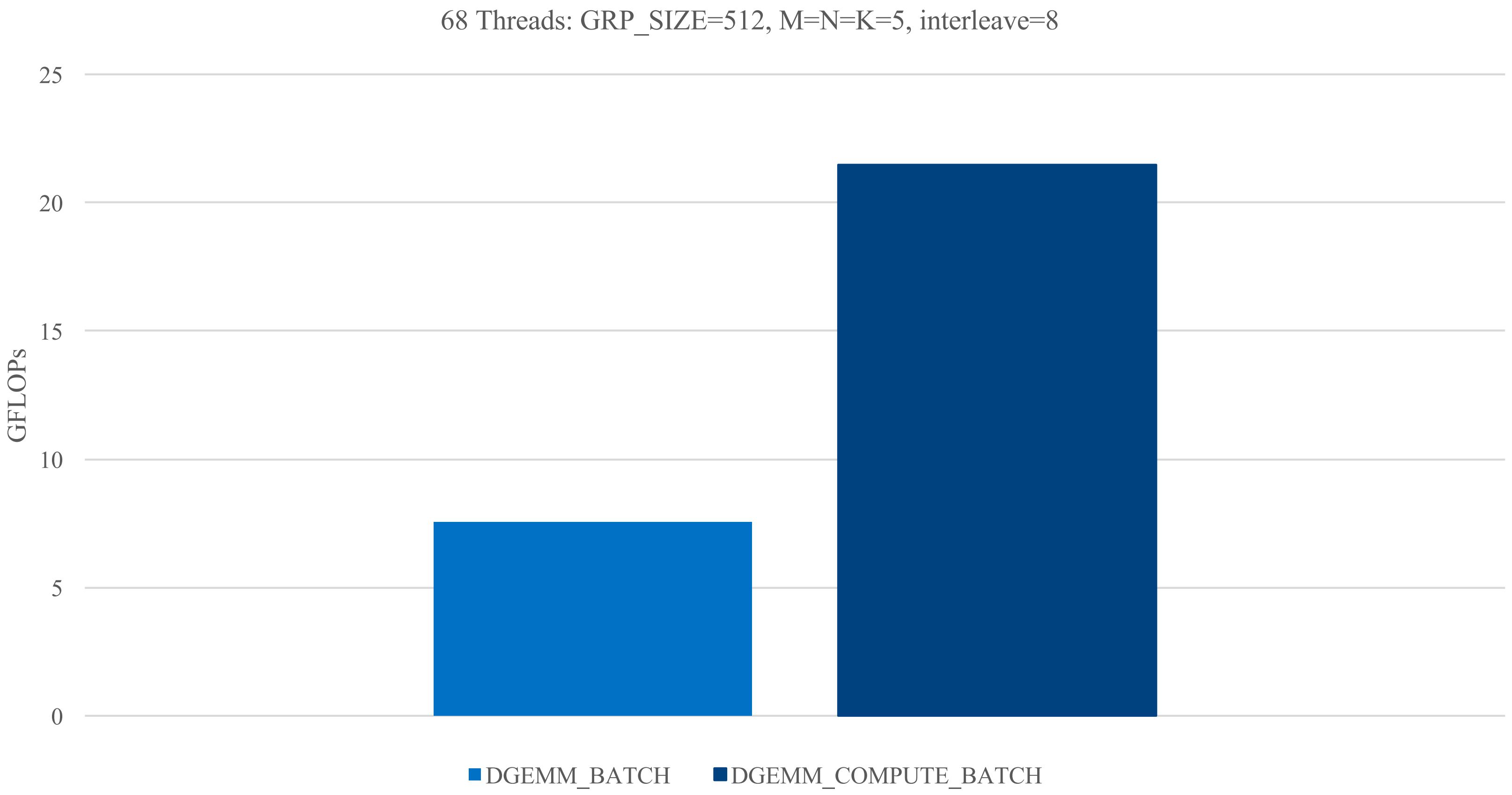
A411	A412	A413
A421	A422	A423
A431	A432	A433

A111	A112	A113
A211	A212	A213
A121	A122	A123
A221	A222	A223
A131	A132	A133
A231	A232	A233

A311	A312	A313
A411	A412	A413
A321	A322	A323
A421	A422	A423
A331	A332	A333
A431	A432	A433

- `if (n_matrices % subgroup_length)` ?
  - Kernels will mask, or users can pad the data.
- Why not fully interleave, i.e. `subgroup_length = n_matrices` ?
  - Spatial locality – elements of matrices will be far apart in memory.

Worth it?



Configuration Info - Hardware: Intel® Xeon Phi™ Processor 7250, 68 cores (34 MB total cache, 1.4GHz), 16GB MCDRAM Memory, 96GB of DDR4 Memory; Operating System: RHEL 7.2 GA x86_64

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. * Other brands and names are the property of their respective owners. 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 .

API Details: Compact Matrix Struct

- API introduces the `compact_t` data type.
- `compact_t` type contains all information for a matrix formatted in the compact API layout:
    - Order, rows, columns, leading dimension, group_count, size_per_group, pointer to data, compact format

- `compact_t mat_p`

Struct containing matrix batch information		
mat_p.rows	MKL_INT_TYPE*	Array of size mat_p.group_count. mat_p.rows(i) gives the number of rows in the group i mat_p matrices.
mat_p.cols	MKL_INT_TYPE*	Array of size mat_p.group_count. mat_p.cols(i) gives the number of columns in the group i mat_p matrices.
mat_p.ld	MKL_INT_TYPE*	Array of size mat_p.group_count. mat_p.ld(i) gives the leading dimension of the mat_p matrices in group i.
mat_p.group_count	MKL_INT_TYPE	Number of groups in the batch of matrices.
mat_p.size_per_group	MKL_INT_TYPE*	Array of size mat_p.group_count. mat_p.size_per_group(i) gives the number of matrices in group i.
mat_p.order	CblasLayout	Set to CblasRowMajor or CblasColMajor. Gives the data layout of the matrices in mat_p.
mat_p.mat	void*	Points to matrix data. Can be set by user who has matrix data formatted according to mat_p.format, or can be allocated and set by functions described in the next section.
mat_p.format	MKL_INT_TYPE	Gives the length of subgroups of matrices that are interleaved. If set to -1, the provided pack function will choose the optimal formatting according to MKL.

API Details: Data Manipulation (skipped by applications already formatting similarly)

- ?BATCH_ALLOC(compact_t* A_p)

Allocates data for batch of partially interleaved matrices. Pointer to allocated data given by A_p->mat

A_p	compact_t*	Parameter struct. Contains matrix information for this matrix batch.

- ?BATCH_PACK(MKL_FP_TYPE** A, compact_t* A_p)

Packs a batch of matrices into an interleaved format

A	MKL_FP_TYPE**	Array of pointers to matrices in standard MKL batched BLAS formatting.
A_p	compact_t*	Parameter struct. Contains matrix information for this matrix batch. Data from A is formatted and stored at A_p->mat.

- ?BATCH_UNPACK(MKL_FP_TYPE** A, compact_t* A_p)

Unpacks a batch of matrices from an interleaved format into standard batched BLAS format

A	MKL_FP_TYPE**	Array of pointers to matrices in standard MKL batched BLAS formatting. Data from A_p.mat is formatted and stored here.
A_p	compact_t*	Parameter struct. Contains matrix information for this matrix batch.

- ?BATCH_FREE(compact_t* A_p)

Frees data allocated by ?BATCH_ALLOC at A_p->mat

API Details: Compute Functions: GEMM

```
?GEMM_COMPUTE_BATCH( CBLAS_TRANSPOSE* TRANSA, CBLAS_TRANSPOSE* TRANSB,
                     MKL_FP_TYPE* alpha, compact_t* A_p, compact_t* B_p,
                     MKL_FP_TYPE* beta, compact_t* C_p )
```

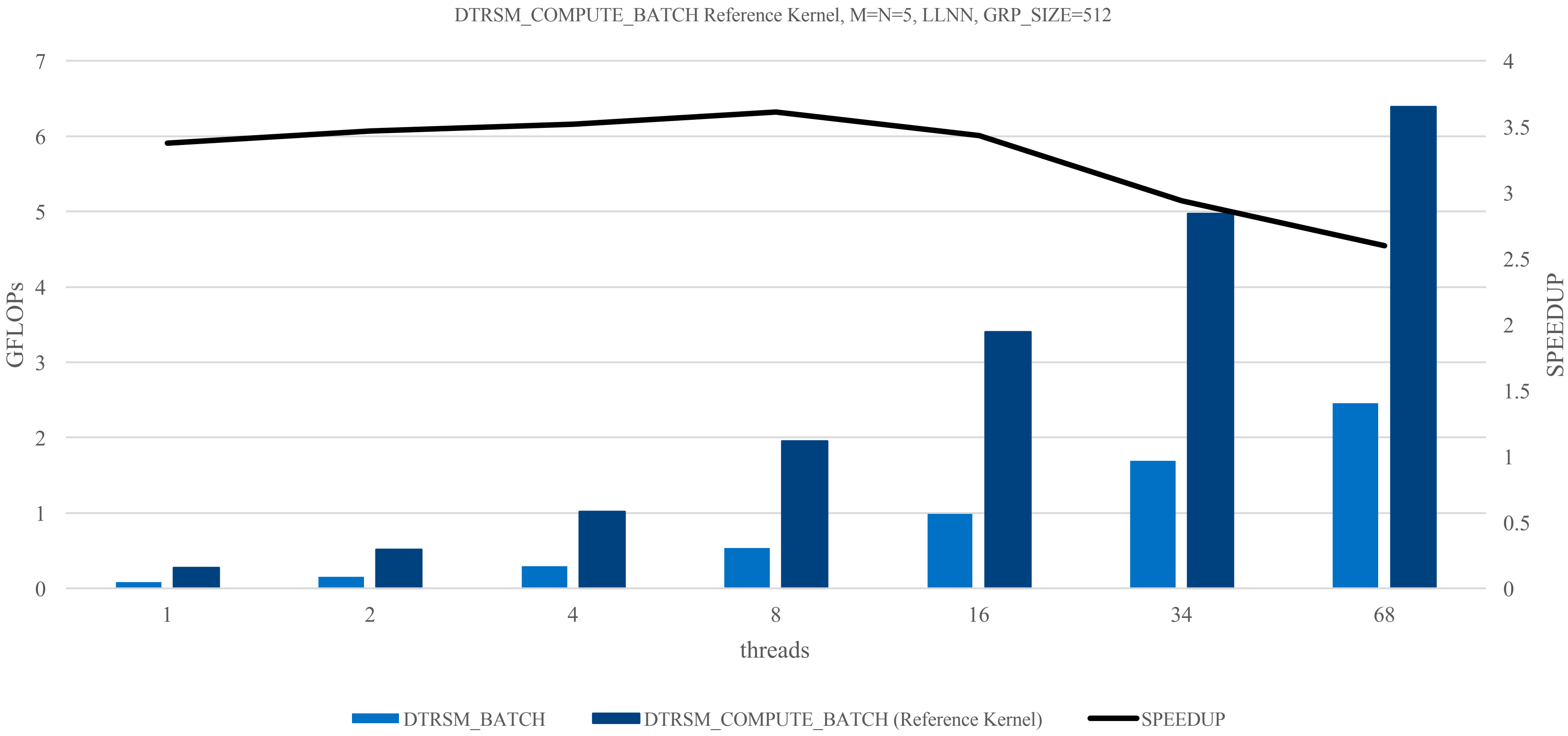
Performs batched GEMM operation on batch of matrices formatted according to A_p, B_p, C_p.		
TRANSA	CBLAS_TRANSPOSE*	Array of size A_p->group_count. TRANSA(i) specifies op(A) for group i.
TRANSB	CBLAS_TRANSPOSE*	Array of size A_p->group_count. TRANSA(i) specifies op(B) for group i.
alpha	MKL_FP_TYPE*	Array of size A_p->group_count. alpha(i) specifies the scalar alpha for group i.
A_p	compact_t*	Parameter struct. Contains matrix information for A matrix batch.
B_p	compact_t*	Parameter struct. Contains matrix information for B matrix batch.
beta	MKL_FP_TYPE*	Array of size C_p->group_count. beta(i) specifies the scalar beta for group i.
C_p	compact_t*	Parameter struct. Contains matrix information for C matrix batch.

API Details: Compute Functions: TRSM

-
  ```
  ?TRSM_COMPUTE_BATCH( CBLAS_SIDE* SIDE, CBLAS_UPLO* UPLO,
                      CBLAS_TRANSPOSE* TRANSA, CBLAS_DIAG* DIAG,
                      MKL_FP_TYPE* alpha, compact_t* A_p,
                      compact_t* B_p)
  ```

Performs batched TRSM operation on batch of matrices formatted according to A_p, B_p.		
SIDE	CBLAS_SIDE*	Array of size A_p->group_count. SIDE(i) specifies whether A is on the left or right of X in group i.
UPLO	CBLAS_UPLO*	Array of size A_p->group_count. UPLO(i) specifies whether A is upper or lower triangular in group i.
TRANSA	CBLAS_TRANSPOSE*	Array of size A_p->group_count. TRANSA(i) specifies op(A) for group i.
DIAG	CBLAS_DIAG*	Array of size A_p->group_count. DIAG(i) specifies whether or not A is unit diagonal in group i.
alpha	MKL_FP_TYPE*	Array of size A_p->group_count. alpha(i) specifies the scalar alpha for group i.
A_p	compact_t*	Parameter struct. Contains matrix information for A matrix batch.
B_p	compact_t*	Parameter struct. Contains matrix information for B matrix batch.

TRSM Reference Kernel Performance:

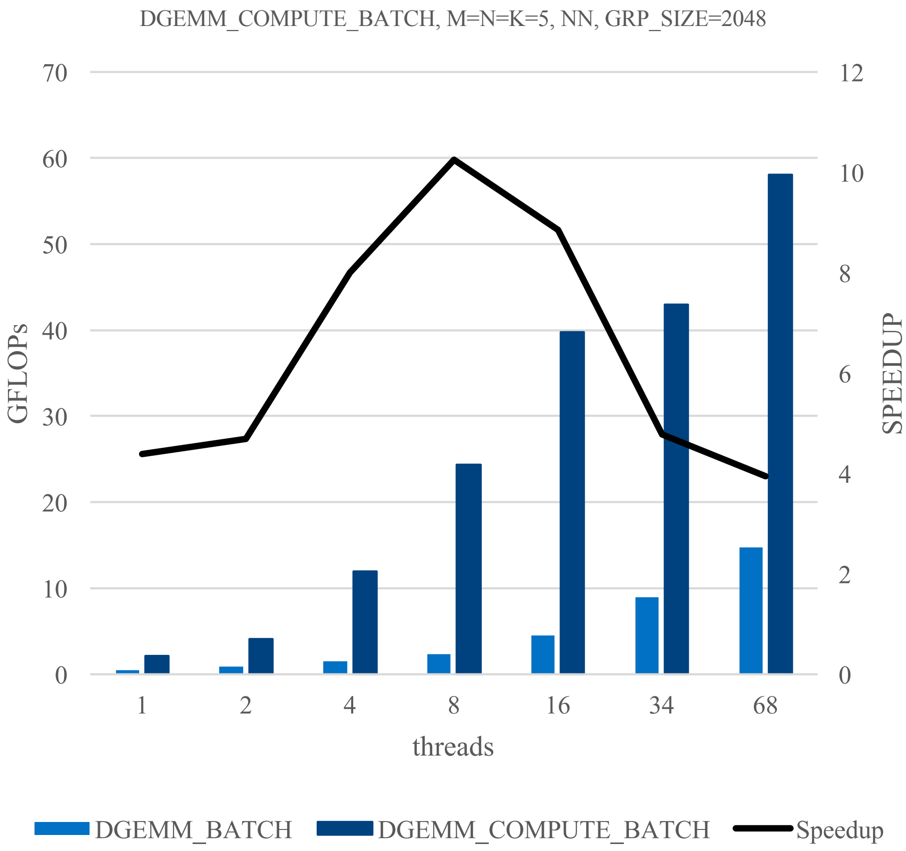


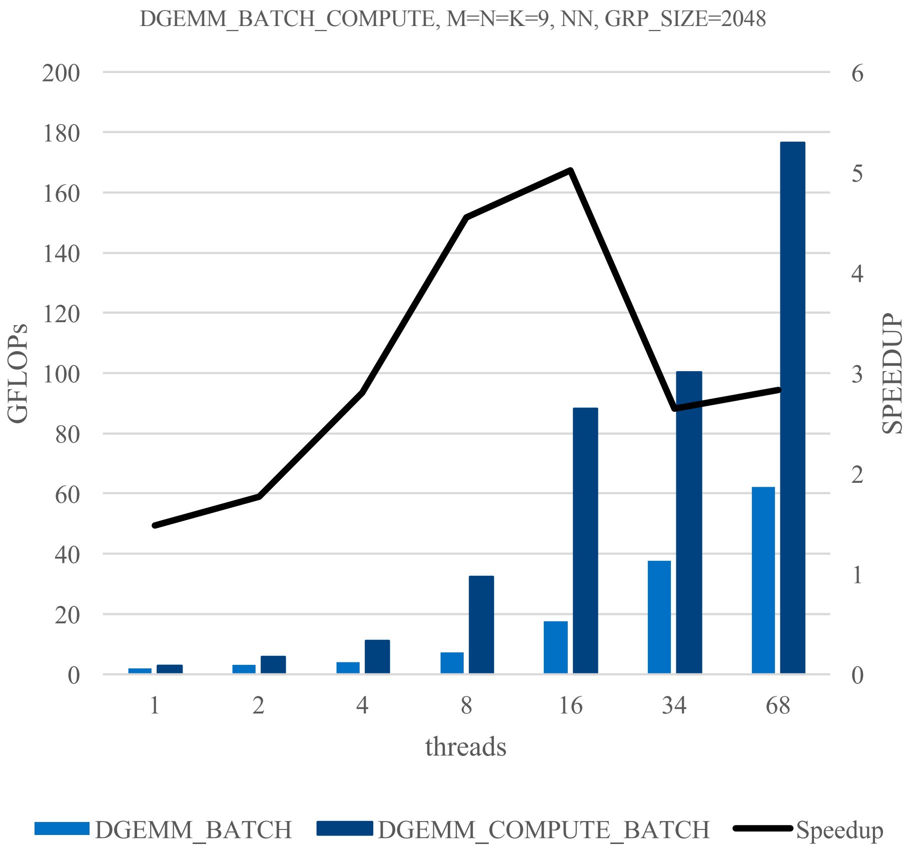
Configuration Info - Hardware: Intel® Xeon Phi™ Processor 7250, 68 cores (34 MB total cache, 1.4GHz), 16GB MCDRAM Memory, 96GB of DDR4 Memory; Operating System: RHEL 7.2 GA x86_64

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. * Other brands and names are the property of their respective owners. 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 .

AVX512 DGEMM NN Prototype Performance:





Configuration Info - Hardware: Intel® Xeon Phi™ Processor 7250, 68 cores (34 MB total cache, 1.4GHz), 16GB MCDRAM Memory, 96GB of DDR4 Memory; Operating System: RHEL 7.2 GA x86_64

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. * Other brands and names are the property of their respective owners. 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 .

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.

Copyright © 2017, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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.

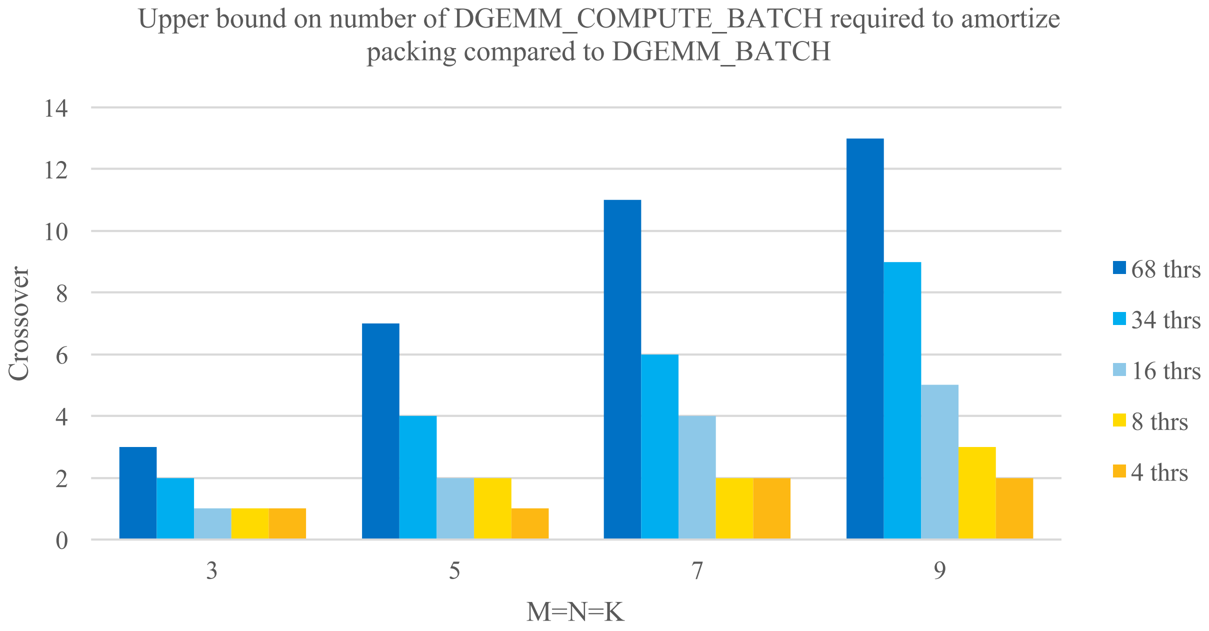
Notice revision #20110804

Backup

Packing Cost

- Current packing function is a serial reference implementation.
- Expect lower cross-over points with optimized implementation.
- Apps that format appropriately will not pay packing cost.
- Cross-over depends on
    - thread count
    - group sizes
    - matrix sizes
    - BLAS operation (e.g. lower cost for TRSM than for GEMM)

- Tests GRP_SIZE=512, DGEMM:



Configuration Info - Hardware: Intel® Xeon Phi™ Processor 7250, 68 cores (34 MB total cache, 1.4GHz), 16GB MCDRAM Memory, 96GB of DDR4 Memory; Operating System: RHEL 7.2 GA x86_64

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. * Other brands and names are the property of their respective owners. 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 .