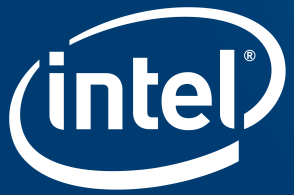


Intel MKL® GEMM_BATCH

Kazushige Goto, Murat E. Guney, *Sarah Knepper*, Shane Story

Intel® Math Kernel Library (Intel® MKL)

Introduction to batched BLAS/LAPACK

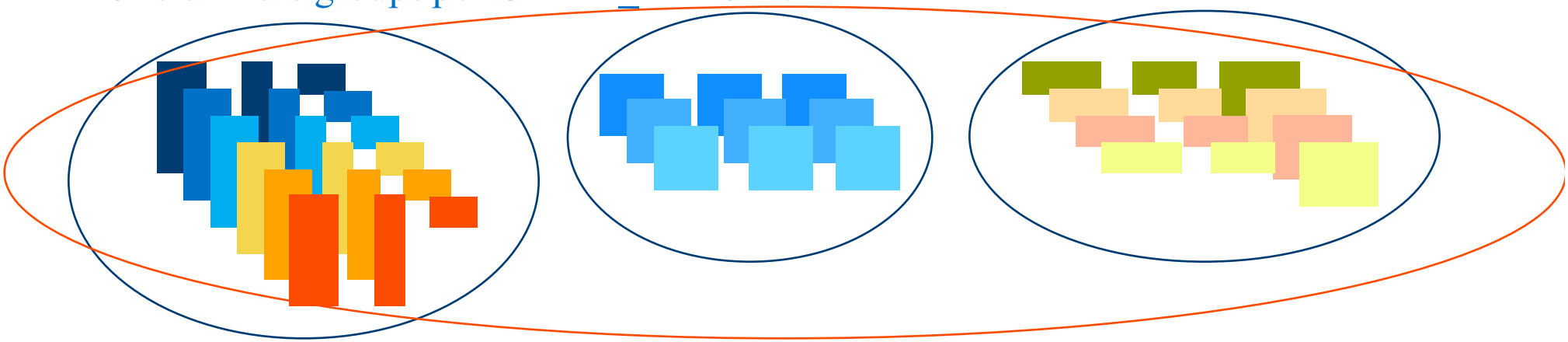
- Execute independent BLAS/LAPACK operations simultaneously with one function call
- User ensures no data dependency between the operations
- Take advantage of all cores even for small/medium sizes
- Existing implementations:
  - cuBLAS*: cublasDgemmBatched
  - MAGMA*: magma_dgemm_batched
  - Intel® Math Kernel Library (Intel® MKL): DGEMM_BATCH

Performance opportunities for batching

- Minimize library overheads for small sizes
  - Function and error checking overheads are significant
  - Dispatch and error check once for each GEMM group
- Better exploit parallelism available in many-/multi-core processors
  - Schedule simultaneous GEMM functions on Intel® Xeon® processor and Intel® Xeon Phi™ coprocessor
  - Assign optimal number of threads/cores to each operation
- Specialized combined-GEMM kernels
  - Vectorization across GEMM calls
  - Prefetch the matrix blocks across GEMM calls

GEMM_BATCH in Intel MKL - Group Concept

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



GEMM_BATCH in Intel MKL - API

- Two additional parameters versus GEMM
  - group_count (integer) : total number of groups
  - group_size (integer*) : number of matrices in each group, array of group_count size
- Consistent level of redirection for function parameters
  - Integer → array of integers
  - Pointer → array of pointers

Example: 100 4x4x4 and 200 2x2x2 SGEMMs

- int group_count = 2;
- int group_sizes[group_count] = {100, 200};
- int m[group_count] = {4, 2};
- int n[group_count] = {4, 2};
- int k[group_count] = {4, 2};
- float *A[300] = {pA0, pA1, pA2, …, pA299};
  - A[0:group_sizes[0]-1]: A_{4x4} matrices for the first group
  - A[group_sizes[0]:group_sizes[1]-1]: A_{2x2} matrices for the second group
- float *B[300] = {pB0, pB1, pB2, …, pB299};
  - B[0:99]: B_{4x4} matrices for the first group
  - B[100:299]: B_{2x2} matrices for the second group

GEMM_BATCH in Intel MKL - CBLAS API

```
void cblas_sgemm_batch (
const CBLAS_LAYOUT Layout,
const CBLAS_TRANSPOSE* transa_array, const CBLAS_TRANSPOSE* transb_array,
const MKL_INT* m_array, const MKL_INT* n_array, const MKL_INT* k_array,
const float* alpha_array,
const float** a_array,
const MKL_INT* lda_array,
const float** b_array,
const MKL_INT* ldb_array,
const float* beta_array,
float** c_array,
const MKL_INT* ldc_array,
const MKL_INT group_count,
const MKL_INT* group_size)
```

Not an array:

- Layout, group_count

Arrays of size group_count:

- transa_array, transb_array, m_array, n_array, k_array, alpha_array, lda_array, ldb_array, beta_array, ldc_array, group_size

Arrays of size sum(group_size[i]):

- a_array, b_array, c_array

Interface of various batched GEMMs versus GEMM

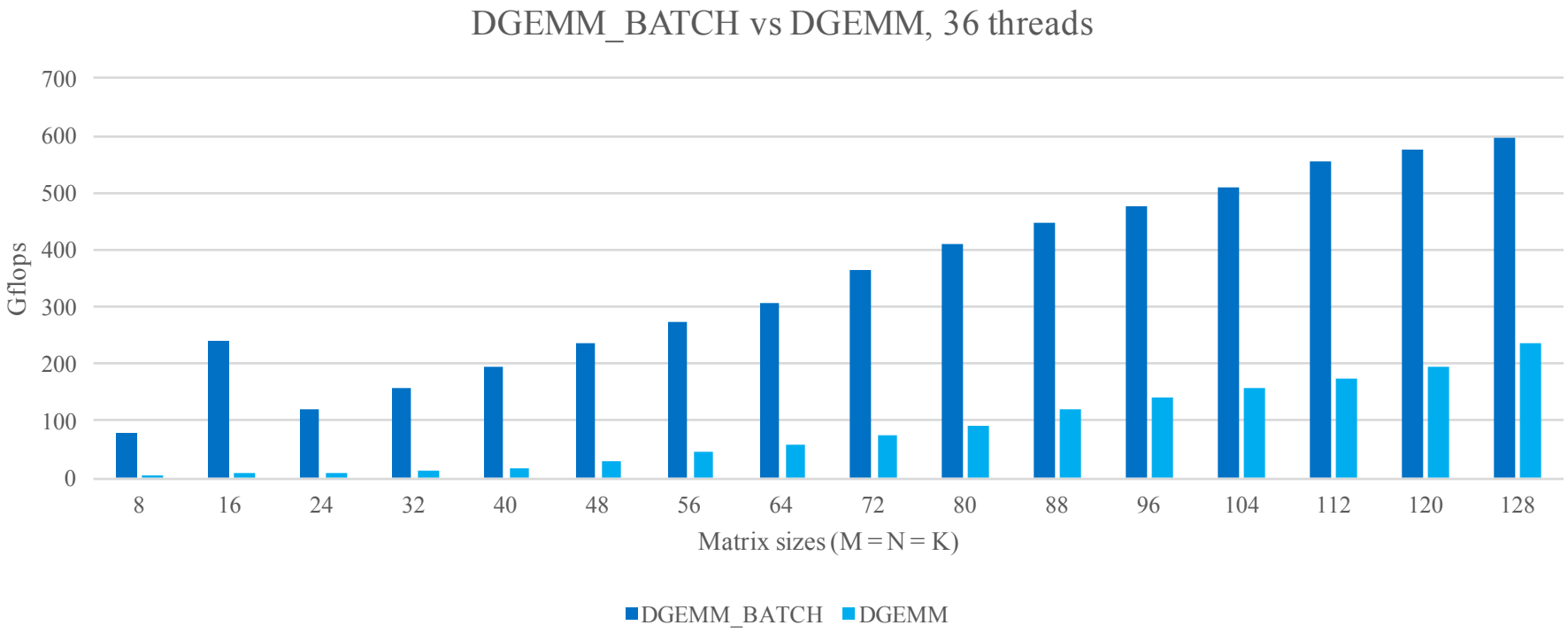
Argument	Description	BLAS sgemm	magma_sgemm_batched	NVidia cublasSgemmBatched	UTK sgemm_batch	Intel MKL sgemm_batch
HANDLE	handle to the cuBLAS library context	--	--	cublasHandle_t	--	--
TRANSA	op(A)	char	char	char	char *	char *
TRANSB	op(B)	char	char	char	char *	char *
M	rows of op(A)/C	int	int	int	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.

GEMM_BATCH in Intel MKL - Error Checking

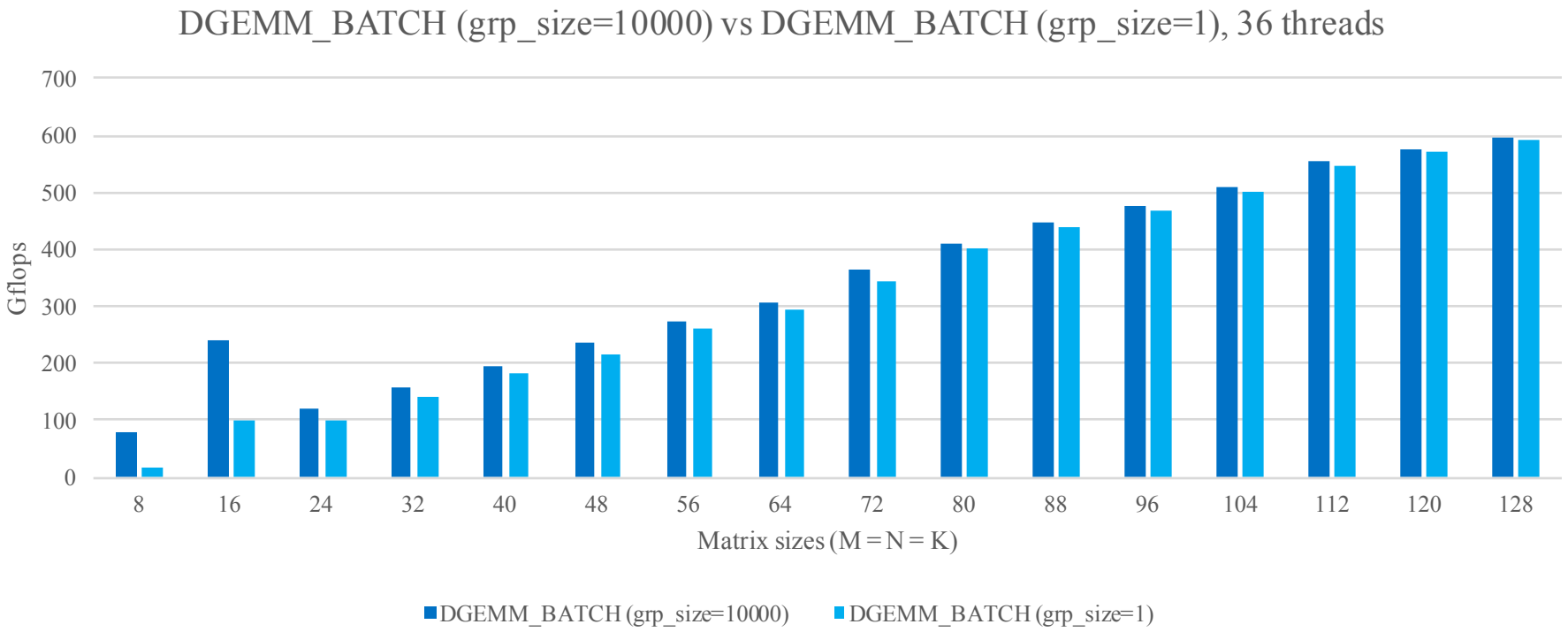
- Similar to existing BLAS/LAPACK error checking
- Input parameters checked before kernel call
  - No computation done if error in any group
- Call XERBLA in case of an error
  - Return which parameter had error, but not which group

DGEMM_BATCH vs DGEMM in Intel MKL
10,000 Matrix Multiplication Instances



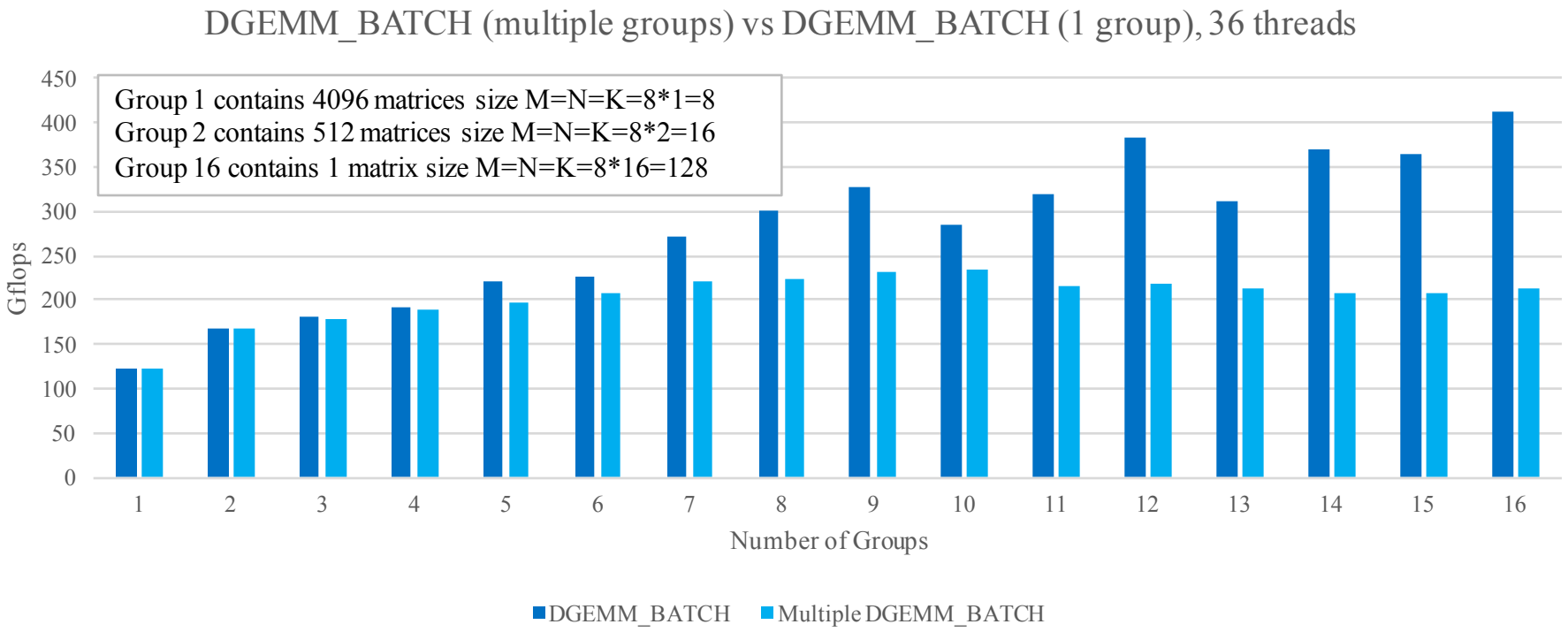
Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.3.3; Hardware: Intel® Xeon® Processor E5-2699v3, 2 Eighteen-core CPUs (45MB LLC, 2.3GHz), 64GB of RAM; Operating System: CentOS 7.1 x86_64

Benefit of Group in DGEMM_BATCH
10,000 Matrix Multiplication Instances



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.3.3; Hardware: Intel® Xeon® Processor E5-2699v3, 2 Eighteen-core CPUs (45MB LLC, 2.3GHz), 64GB of RAM; Operating System: CentOS 7.1 x86_64

Benefit of Multiple Groups in DGEMM_BATCH Similar FLOP count per group



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.3.3; Hardware: Intel® Xeon® Processor E5-2699v3, 2 Eighteen-core CPUs (45MB LLC, 2.3GHz), 64GB of RAM; Operating System: CentOS 7.1 x86_64

Final Remarks

- Batching better utilizes multi-/many-cores for small/medium matrices
- Groups contain matrices with same parameters (size, leading dimension, etc.)
- Intel MKL GEMM_BATCH API allows batching multiple groups
  - Reduces overhead of function calls
  - Minimizes parameter checking
  - Opportunities for cross-GEMM optimizations
- Intel MKL GEMM_BATCH API combines ease-of-use with performance opportunities

Legal Disclaimer & Optimization Notice

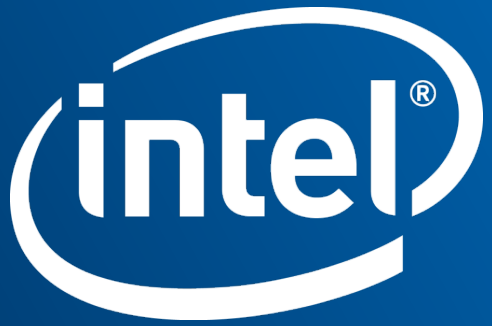
INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING 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