جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

KBLAS: High Performance Level-2 BLAS on Multi-GPU Systems

Ahmad Abdelfattah, David Keyes, and Hatem Ltaief

Center of Extreme Computing, King Abdullah University of Science and Technology

Introduction

KBLAS (KAUST BLAS) is a small library that provides highly optimized BLAS routines on systems accelerated with GPUs. KBLAS is entirely written in CUDA C, and targets NVIDIA GPUs with compute capability 2.0 (Fermi) or higher. The current focus is on level-2 BLAS routines, namely the general matrix vector multiplication (GEMV) kernel, and the symmetric/hermitian matrix vector multiplication (SYMV/HEMV) kernel. KBLAS provides these two kernels in all four precisions (s, d, c, and z), with support to multi-GPU systems. Through advanced optimization techniques that target latency hiding and pushing memory bandwidth to the limit, KBLAS outperforms state-of-the-art kernels by 20-90% improvement. Competitors include CUBLAS-5.5, MAGMABLAS-1.4.0, and CULA-R17. KBLAS scores at least 75% of the sustained peak performance on a Kepler K20c GPU. The SYMV/HEMV kernel from KBLAS has been adopted by NVIDIA, and should appear in CUBLAS-6.0

KBLAS Kernels

KBLAS currently implements the following standard BLAS operation: $\mathbf{y} = \alpha \mathbf{A}\,\mathbf{x} + \beta\,\mathbf{y}$, where α and β are scalars, x and y are one dimensional vector each, and A can be a symmetric/hermitian matrix (SYMV/HEMV) kernel, or a general non-symmetric matrix (GEMV). KBLAS supports all four precisions (single, double, complex, and complex double). It builds on top of two kernels that have been proposed before to the HPC community [1][2]. However, the current versions have been standardized to fully comply with the standard BLAS interface. For example, the SYMV kernel proposed in [1] used to require an extra workspace as an input. The current SYMV, however, makes use of atomic operations to waive such requirement. The current GEMV performance has been made smoother, thanks again to the improved atomic operations, which enable multiple thread blocks to share the same output vector segment, thus increasing the level of parallelism on the GPU

Single GPU Performance

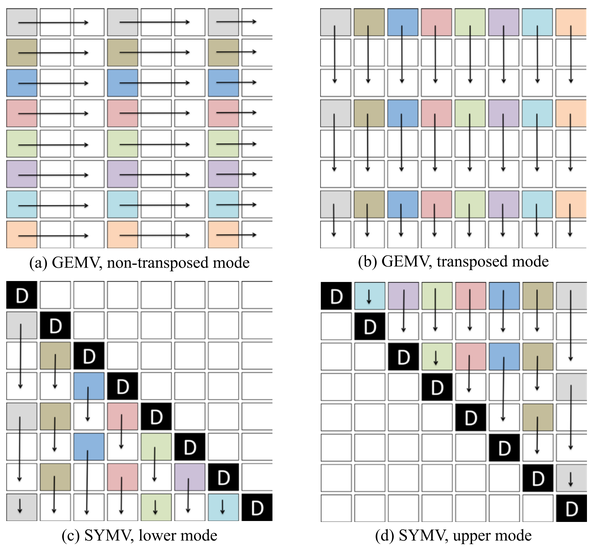
- Competitors
  - CUBLAS-5.5, MAGMABLAS-1.4.0, and CULA-R17. All libraries are tested under CUDA-5.5.
- GPU Architecture: Kepler K20c, ECC on
- xGEMV kernel (Figure 4):
  - Competitive performance
  - Slight advantage for the Z-precision.
  - Scored performance is 97% close to the sustained peak.
  - CUBLAS outperform KBLAS for tall and skinny matrices
- xSYMV/xHEMV (Figure 5):
  - Asymptotic improvement against the best competitor (MAGMABLAS) is 30% to 42%
  - Against CUBLAS-5.5, it is 30% to 57%

Multi-GPU Performance

- Competitors
  - MAGMABLAS-1.4.0 (SYMV/HEMV only), tested under CUDA-5.5.
- GPU Architecture: 1-8 Kepler K20c GPUs (ECC off)
- xGEMV (Figure 6):
  - Only provided by KBLAS
  - Performance oscillations start to appear on 5 GPUs and beyond
  - Asymptotic performance for large matrices is stable
- xSYMV/xHEMV (Figure 7):
  - Asymptotic improvement against MAGMABLAS is between 40% and 90%
  - Atomic operations play a key role in achieving high performance for relatively small matrices

Design Approach: Grid Level

- Matrix processed in square tiles
- Multiple thread blocks write to the same output vector segment using atomics (improvement over older versions [1][2])
- In the symmetric case, diagonal tiles are processed in a separate kernel, since they have special computation pattern
- Figure 1 summarizes the grid-level design for all kernels



(a) GEMV, non-transposed mode (b) GEMV, transposed mode

(c) SYMV, lower mode (d) SYMV, upper mode

Figure 1: Matrix traversal using CUDA thread-blocks

Design Approach: Thread-block Level

- Summarized in Figure 2
- Data prefetching through double buffers using only registers
- Restricted shared memory role to avoid frequent synchronization
- On-the-fly computation for the transposed off-diagonal computation in the symmetric case (i.e. no need to transpose the tile)

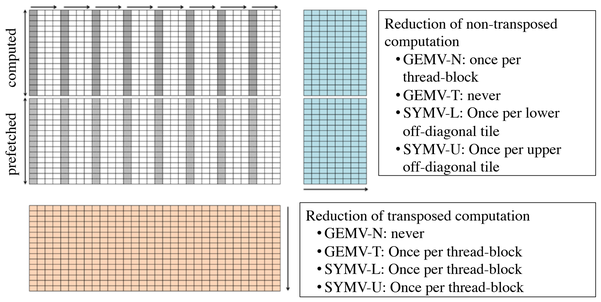


Figure 2. Processing of a matrix tile in KBLAS GEMV/SYMV

Porting on Multiple GPUs

- GEMV/SYMV kernels on multi-GPU are important for huge matrix that do not fit into single GPU memory
- The matrix has to be distributed among GPUs
- The 1D cyclic block-column approach is very convenient to support standard higher-level LAPACK routines (e.g. matrix reduction into bidiagonal/tridiagonal forms). Such layout choice was proposed by MAGMABLAS [3]
- KBLAS uses the same format (Figure 3)

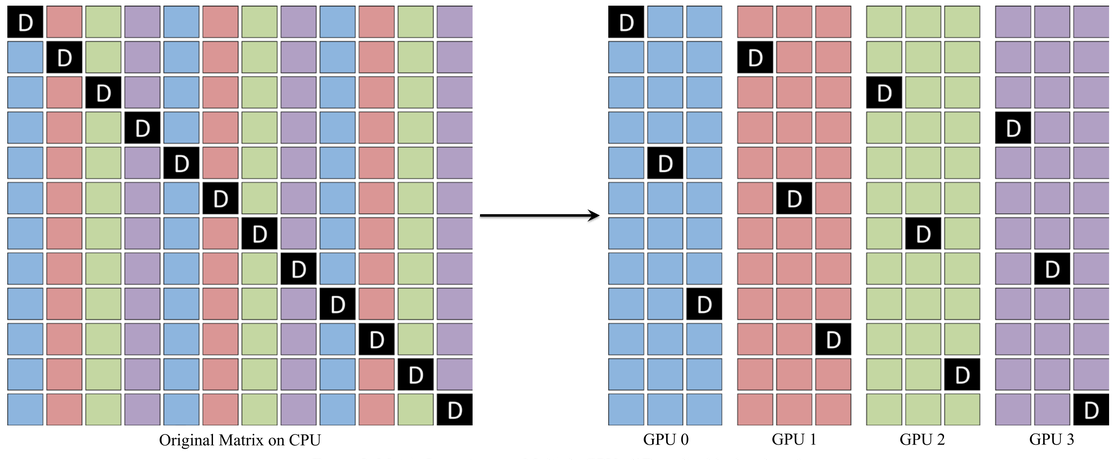


Figure 3. Matrix Layout across Multiple GPUs (1D cyclic, block column)

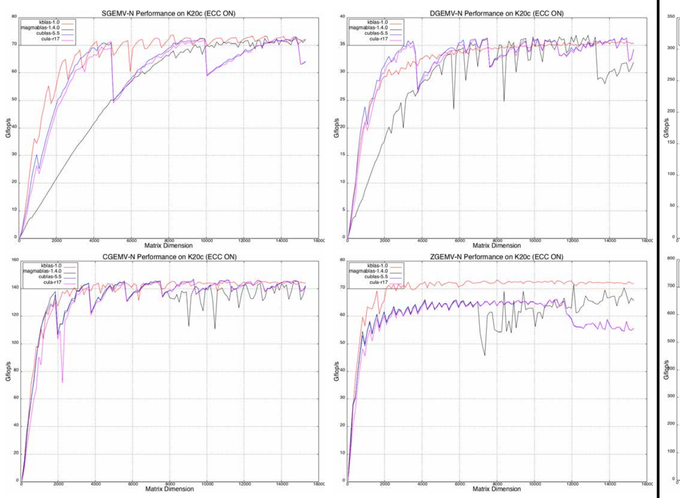


Figure 4: Performance of KBLAS GEMV against CUBLAS, MAGMABLAS, and CULA

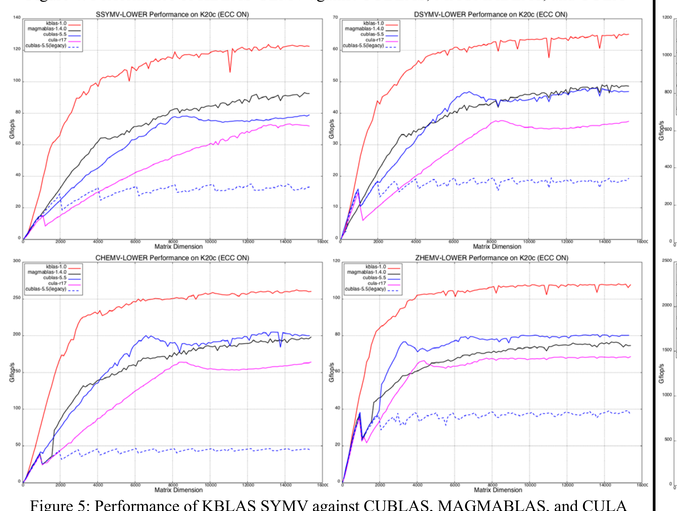


Figure 5: Performance of KBLAS SYMV against CUBLAS, MAGMABLAS, and CULA

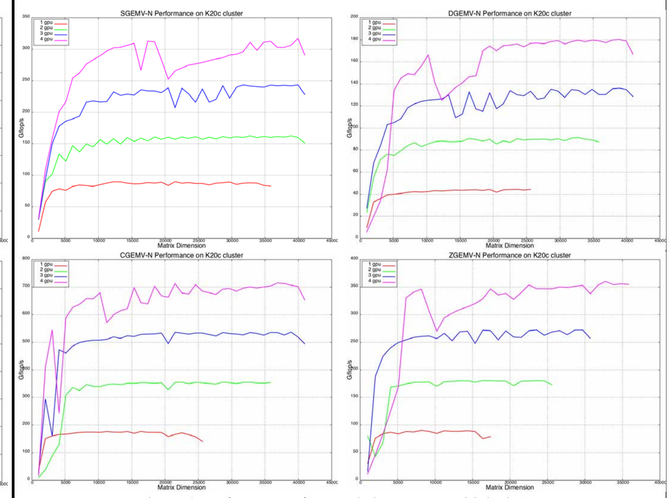


Figure 6: Performance of KBLAS GEMV on multiple GPUs

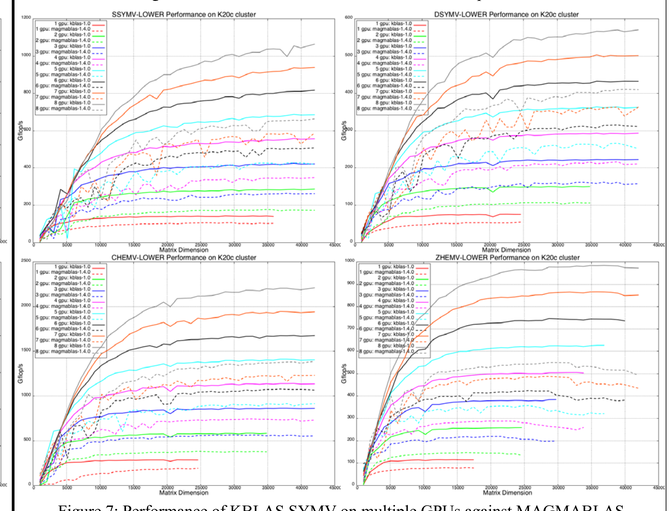


Figure 7: Performance of KBLAS SYMV on multiple GPUs against MAGMABLAS

Acknowledgement

We thank Philippe Vandermersch and Sharan Chetlur from NVIDIA for their support during the integration of the KBLAS SYMV/HEMV kernel into CUBLAS-6.0.

Download

To download KBLAS, please visit this webpage:
http://cec.kaust.edu.sa/Pages/Abelfatteh.aspx

References

[1] Abdelfattah, A., Dongarra, J., Keyes, D., Ltaief, H. "Optimizing Memory-Bound Numerical Kernels on GPU Hardware Accelerators," in The 10th International Meeting on High Performance Computing for Computational Science (VECPAR), 2012

[2] Abdelfattah, A., Keyes, D., Ltaief, H. 2012. Systematic approach in optimizing numerical memory-bound kernels on GPU. In Proceedings of the 18th international conference on Parallel processing workshops (Euro-Par'12)

[3] Yamazaki, I., Dong, T., Solca, R., Tomov, S., Dongarra, J., Schulthess, T. "Tridiagonalization of a dense symmetric matrix on multiple GPUs and its application to symmetric eigenvalue problems," Concurrency and Computation: Practice and Experience, Oct. 2, 2013