Enabling High Performance Large Scale Dense Problems through KBLAS

Ahmad Abdelfattah, David Keyes, and Hatem Ltaief
Extreme Computing Research Center (ECRC), 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 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. The SYMV/HEMV kernel from KBLAS has been adopted by NVIDIA, and should appear in CUBLAS-6.0. KBLAS has been used in large scale simulations of multi-object adaptive optics.

KBLAS Kernels

KBLAS currently implements the following standard BLAS operation:

\[ y = \alpha x + \beta y \]

- A can be general (GEMV), or symmetric/Hermitian (SYMV/HEMV)
- All four precisions are supported (single, double, complex, and complex double)
  - Single precision (SGEMV – SSYMV)
  - Double precision (DGEMV – DSYMV)
  - Complex precision (CGEMV – CHERMV)
  - Complex double precision (ZGEMV – ZHEMV)

Fully compliant with standard BLAS API interface

Current version builds on top of previously proposed work [1][2]

These kernels run on both one GPU and multi-GPU, as long as GPUs share the same host (i.e., single node)

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-chip 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

Design Approach: Grid Level

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

Performance Summary

- Consider real double precision as an example
  
  - CUBLAS-5.5, MAGMABLAS-1.4.0, and CULA-R17. All libraries are tested under CUDA-5.5.
  
  - GPU Architecture: Kepler K20c, ECC off
  
  - Single GPU Performance (Kepler K20c, ECC off): Figure 4
  
  - DGEMV Performance:
    - Very similar asymptotic performance
    - Smoother behavior for relatively small matrices
  
  - DSYMV Performance:
    - 44% against MAGMA BLAS-1.4.0
    - 56% against NVIDIA’s CUBLAS
    - 97% against CULA-R17

- Multi-GPU (up to 8 Kepler K20c’s, ECC off): Figure 5
  
  - DGEMV:
    - Only KBLAS provides DGEMV on multi-GPU
    - Oscillatory behavior currently under study
  
  - Performance improvement of DSYMV:
    - Up to 40% against MAGMABLAS
    - CUBLAS and CULA do not support similar functionality

Application Case Study: Multi-object Adaptive Optics (MOAO)

- In collaboration with Observatoire de Paris
- High performance simulation of pseudo-analytical model for multi-object spectrographs (MOS)

- The output of the study helps the design of a MOS proposed for the Extremely Large Telescope (E-ELT)[4]

- Concept is illustrated in Figure 6:
  
  - IFU (Integre Field Unit) is the optical setup that will split the tiny image of the observed galaxy into pixels which light will be dispersed in the spectrograph
  
  - 20 IFUs across the field of View (FoV)
  
  - 10 WFS (Wavefront Sensors) across the field, each with 4000 measurements

- Model involves pseudo-inversion of large dense matrix (40k x 40k) in double precision

- Inversion is based on eigenvalue decomposition (possibly filtering out some eigenvalues)

- KBLAS DSYMV kernel is used to accelerate the eigen solver (DSYEVD) from MAGMA, which accounts for ~90% of total run time

- DSYEVD is improved (against the best competitor), by 35% on single GPU, and up to 70% on 8 GPUs

Figure 6: Concept of Multi-object Adaptive Optics

References


Figure 7: Eigen solver performance on single and multi-GPU. Accelerated MAGMA uses KBLAS DSYMV instead of MAGMABLAS DSYMV

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

KBLAS-1.0 is available for download through this webpage:
http://ecc.ku.edu.sa/ Pages/Abdelfattah.aspx