AOCL 3.2

Meghana Vankadari

AMD

Agenda

Introduction to AOCL

Optimizations

New Features

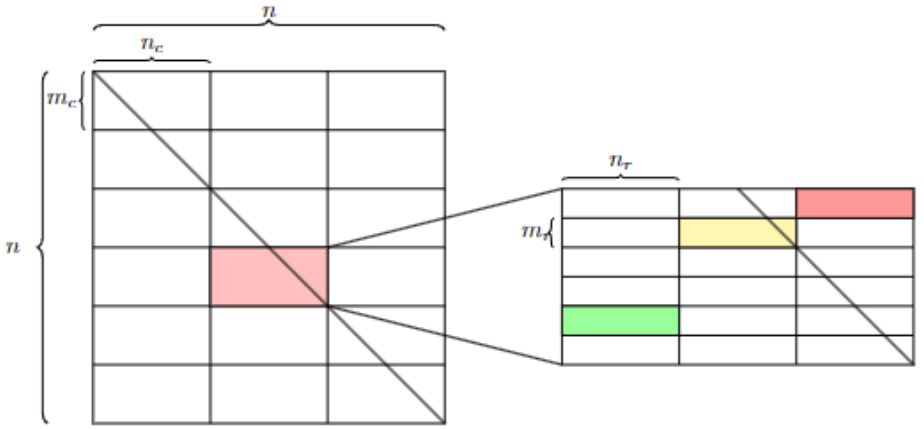
Q&A

Introduction

- AOCL – AMD optimizing CPU Libraries
  - AOCL are a set of numerical libraries tuned specifically for AMD EPYC™ processor family
  - BLIS is part of this AOCL
  - https://developer.amd.com/amd-aocl/
  - Latest source code for BLIS is available on GitHub https://github.com/amd/blis
- For any issues or queries regarding the libraries, please contact aoclsupport@amd.com
- Latest version is AOCL 3.2
  - Includes optimizations for “Zen 3”
  - GEMMT, Dynamic Parallelization and DGEMM

GEMMT

- Consider a lower-triangular gemmt problem, the execution of top 4 loops(jc, kc, ic, jr) is same as GEMM
- In the ‘ir’ loop, the execution for each block of c differs based on the placement of the particular block w.r.t diagonal
- SUP for gemmt is implemented to handle smaller sizes



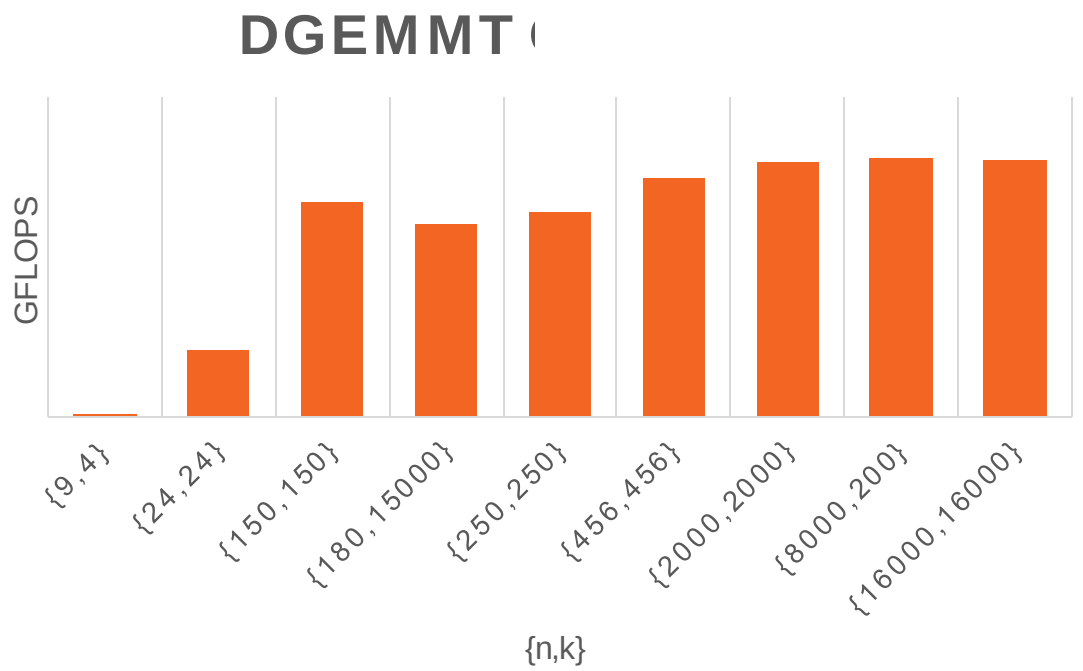
If present below the diagonal, call gemm kernel

If present along the diagonal, use an extra buffer to store alpha*A*B and copy only the lower triangular part to C later

If present above the diagonal, ignore

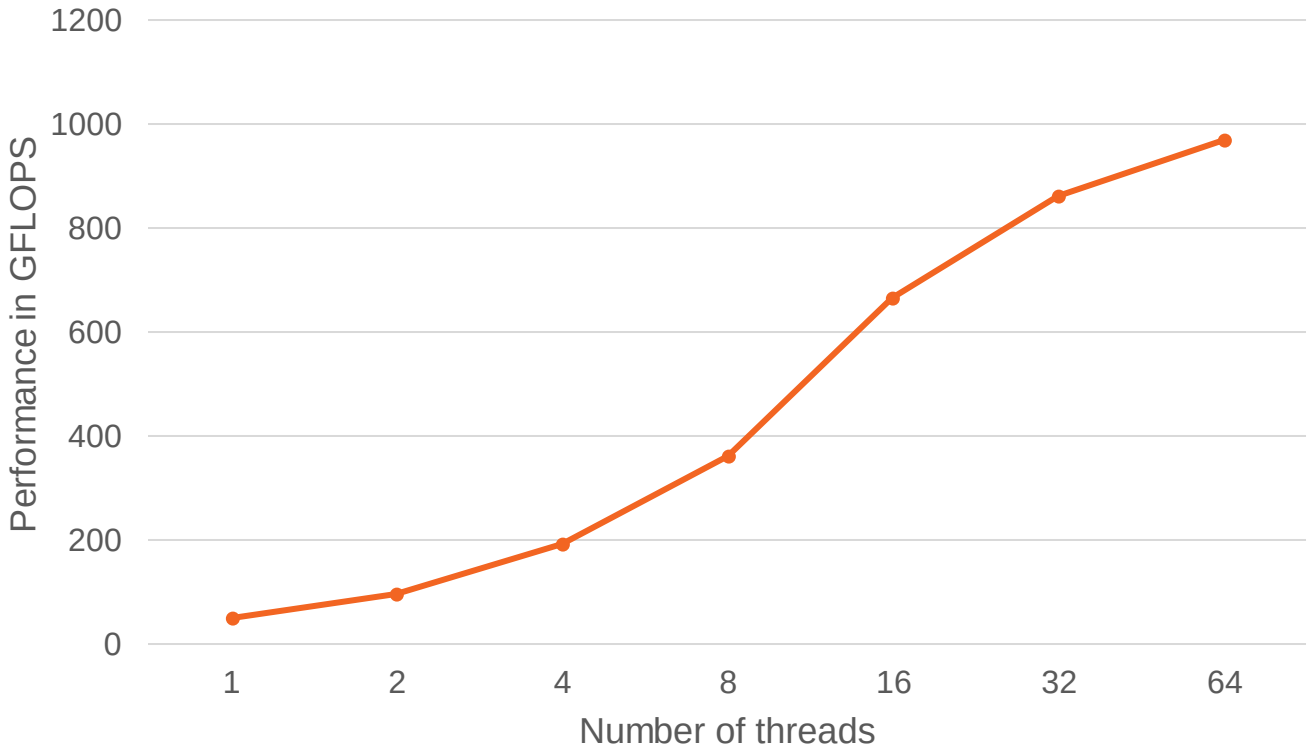
DGEMMT-ST performance

Below data is captured for the case where C is lower triangular matrix and alpha and beta are !=0 and !=1



DGEMMT-MT performance

Since C matrix is triangular, GEMMT uses weighted thread partitioning to divide workloads among threads.

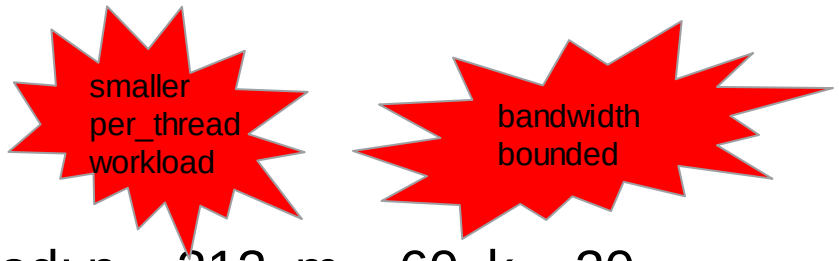


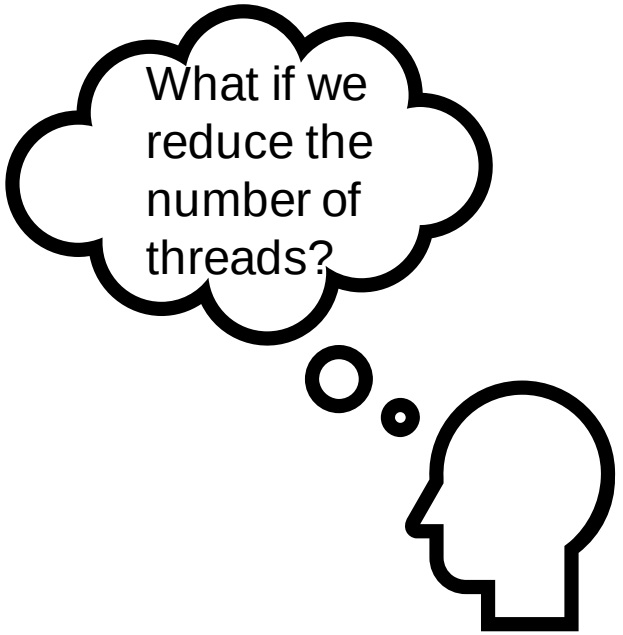
Parallelization in BLIS

Consider an example of application calling GEMM with following matrix dimensions.

n= 20,000, m = 60, k = 20, row-major order

num_threads = 64 ➔ jc_nt = 64





Workload per thread: n = 312, m = 60, k = 20

performance: 12.379 GFLOPS

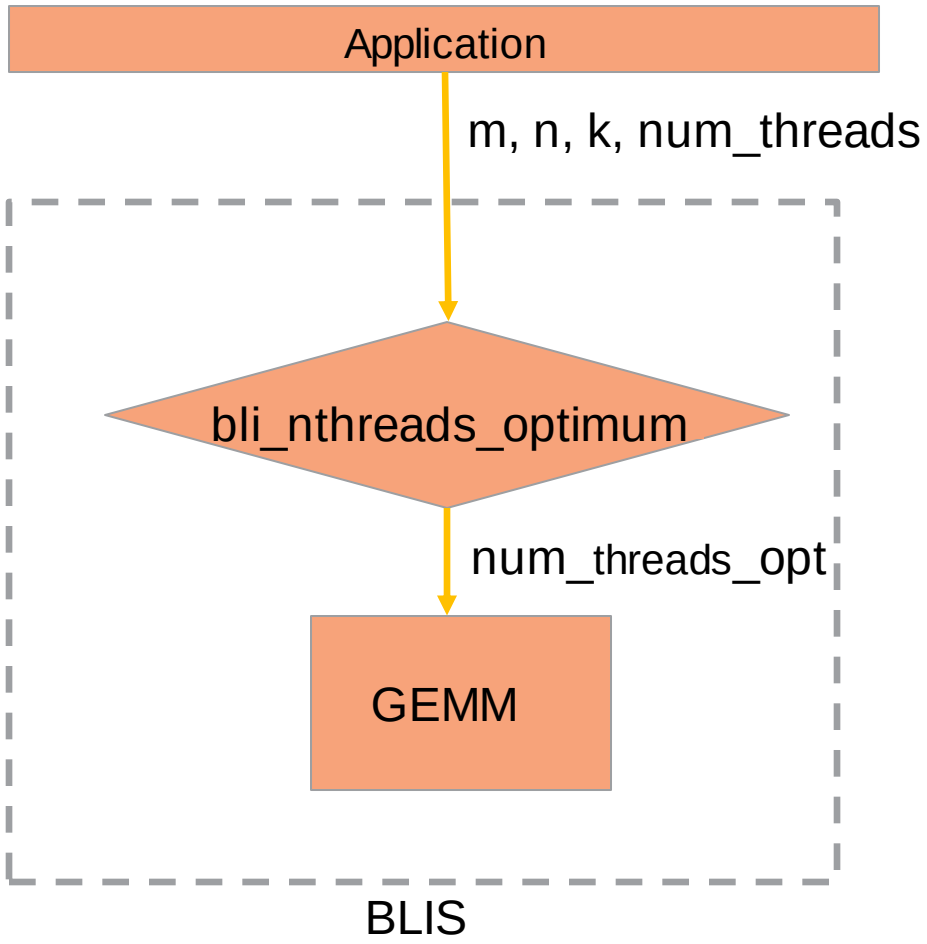
num_threads=16 ➔ jc_nt = 16

workload per thread: n = 1256, m = 60, k = 20

performance: 43.302 GFLOPS

Dynamic Parallelization

- Dynamic parallelization improves multi-threaded performance of BLIS by calculating optimal number of threads for a given input
- Bli_nthreads_optimum() determines optimal number of threads using dimensions of matrices passed by the application – m, n, k
- Available for
  dgemm, zgemm, dsyrk, dtrsm, ztrsm, dgemmt, dtrmm



Dynamic parallelization performance report

m	n	k	Num threads default	Default Perf (GFLOPS)	Num threads optimum	Perf with dynamic threading (GFLOPS)	%gain
2000	700	60	64	261.889	16	626.999	139.4
20000	60	10	64	12.379	16	43.302	249.8
100	10	256	64	26.126	8	50.300	92.5
200	40	500	64	261.028	32	393.740	50.8
700	8	72	64	32.652	4	56.246	72.2
30	20	24	64	1.458	1	22.821	1465.2

DGEMM

- The GEMM operation is defined as

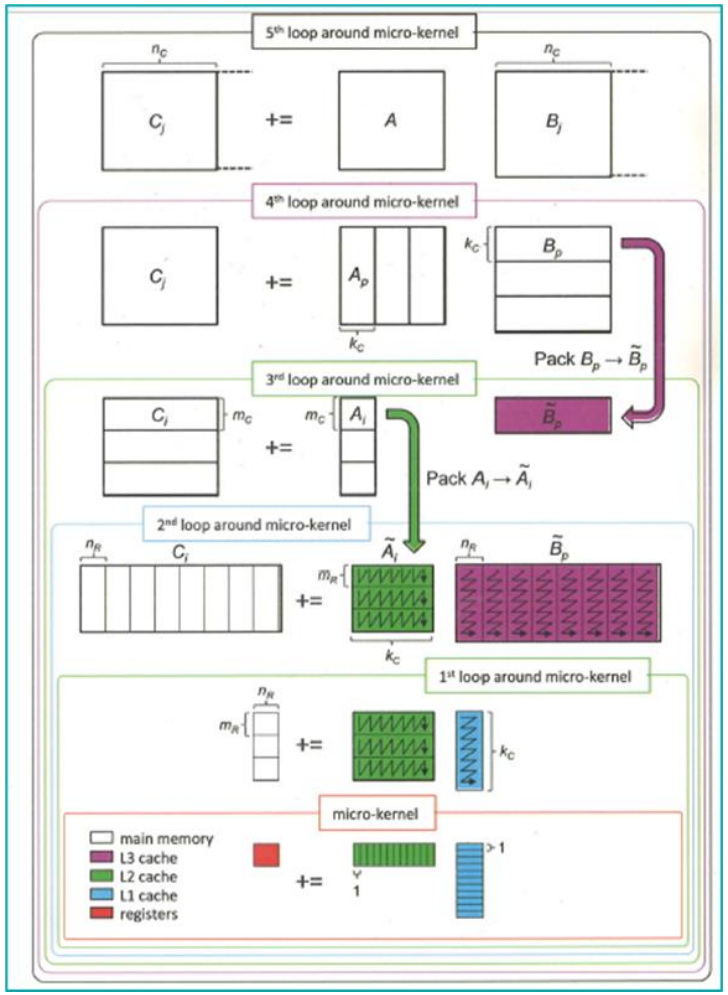
  C = beta∗C + alpha∗op(A) ∗ op(B)

  Where:

  Op(X)=X or op(X)=X^T or op(X)=X^H alpha and beta are scalars A, B and C are matrices:

  Op(A)is an m ×k matrix , Op(B) is an k ×n matrix C is an m×n matrix

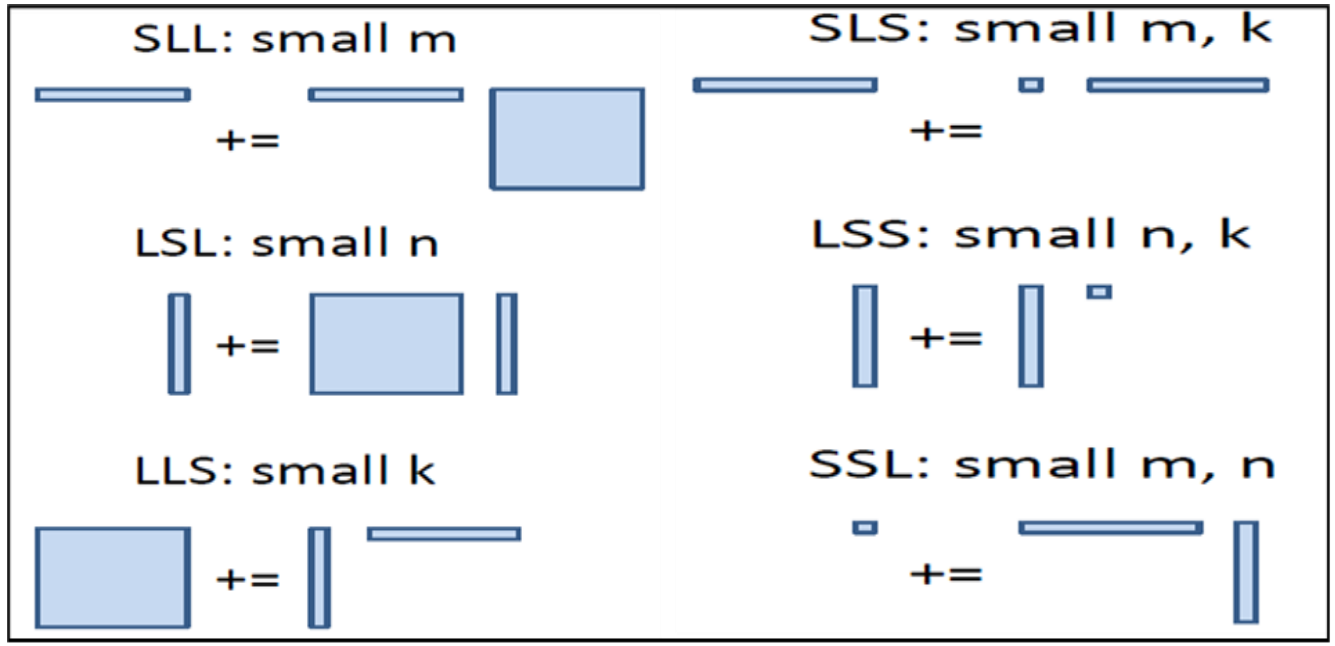
- The performance of GEMM primarily depends on shapes of the matrices defined by dimension of the matrices (m, k, n)
- To achieve consistent performance across wide spectrum of matrix sizes, BLIS primarily employs multiple GEMM implementations



Native GEMM algorithm in BLIS

Source: Huang, Jianyu & van de Geijn, Robert. (2016). BLISlab: A Sandbox for Optimizing GEMM.

Matrix shapes



Source: https://www.cs.utexas.edu/users/flame/BLISRetreat2019/slides/Field.pdf

Six Shape Scenarios of Skewed Matrices based on (mnk)
S: small, L: Large

Examples of Matrix shapes

LSS

m	n	k	lda	ldb	ldc
6288	128	126	8202	480	9462
840	128	126	8202	480	9462
1864	128	126	8202	480	9462

SSS

m	n	k	Op(A)	Op(B)
128	128	128	n	n
128	126	126	t	n
128	128	126	t	n

Optimizations

- For large sizes – native DGEMM Implementation
- For skinny sizes – packing is expensive
  - We use “sup” (small unpacked DGEMM implementation)
  - For LSS matrix shape – matrix A is large, and Matrix B is small
  - After internal transpose, A becomes small and B becomes large
  - Since B is large, block-panel(var2m) implementation works effectively compared to panel-block(var1n)
  - Matrix A is L2 cache-blocked and matrix B is L3 blocked

m	n	k	lda	ldb	ldc	perf with var1n	perf with var2m
6288	128	126	8202	480	9462	39.94	43.44
840	128	126	8202	480	9462	40.26	46.05
1864	128	126	8202	480	9462	40.24	46.10

Optimizations - continued

- For SSS – Developed dgemm_small
  - Does only register blocking. No cache blocking
  - Uses 16x3 kernel maximizing the SIMD usage along larger dimension
  - Kernel is col-preferred
  - Works efficiently for small and skinny sizes

m	n	k	Op(A)	Op(B)	BLIS_3.0	BLIS_3.2
128	128	128	n	n	42.88	48.71
128	126	126	t	n	43.75	48.65
128	128	126	t	n	43.55	48.50

BLAS-like extensions

- **Developed new BLAS-like extension APIs**
  - Copy routines:
    - ?imatcopy: Performs scaling and in-place transposition/copying of matrices
    - ?omatcopy: Performs scaling and out-of-place transposition/copying of matrices
    - ?omatadd: Performs scaling and sum of two matrices including their out-of-place transposition/copying
    - ?omatcopy2: Performs two-strided scaling and out-of-place transposition/copying of matrices
  - GEMMT, Gemm3m and gemm_batch, dzgemm
  - Level-1:
    - amin: Finds the index of the element with the smallest absolute value
    - axpby: Scales two vectors, adds them to one another and stores result in the vector
  - Supports CBLAS and BLAS interfaces.

questions?

DISCLAIMER AND ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.


AMD