Towards Performance Portability of Highly Parametrizable TRSM Algorithm Using SYCL

Thales Sabino
Mehdi Goli

IWOCL-SYCL 2021
Agenda

• Introduction
• The TRSM Problem
• GEMM-Based TRSM
• Performance Evaluation
• Conclusion and Future Work
Introduction

• **TRiangular Solve with Multiple Right-Hand Sides**
• TRSM is an important operation used to solve linear systems efficiently
• Performance portable GEMM-based TRSM solver in SYCL-BLAS
• TRSM is simple to solve sequentially
• Can be solved in parallel devices using a GEMM-based approach
Motivations

• Accelerated BLAS routines

• Architectures
  • CPU/GPU/FPGA
  • Embedded Accelerators

• Performance portability
  • Library approach
    • BLAS/ MKL/cuBLAS
  • Parallel pattern abstractions
    • Raja/Kokkos/Eigen

• Challenges
  • Provide a cross-platform performance portable programming model for future development
SYCL

• C++ based open standard API introduced by Khronos
• Provides single-source programming model for accelerators
• Provides an implicit execution graph by tracking kernel dependencies
• Multiple implementations
  • ComputeCpp
  • Intel DPC++
  • hipSYCL
  • triSYCL
• Programming model
  • Kernel Scope
  • Command group scope
  • Application Scope
The TRSM Problem

• Triangular solve with multiple right-hand sides
• Solve for $X$ in one of the following matrix equations:

$$\text{op}(A)_{(m,m)}X_{(m,n)} = \alpha B_{(m,n)}$$
$$X_{(m,n)}\text{op}(A)_{(n,n)} = \alpha B_{(m,n)}$$

• $\text{op}(A) = A$ or $\text{op}(A) = A^T$
• $A$ can be upper or lower triangular
• $A$ can have a unit or non-unit diagonal
• $A$ can be on the left or right side of $X$
• $\alpha$ is a scalar
Solving a TRSM Problem Sequentially

\[
\begin{bmatrix}
  a_{00} & a_{01} & a_{02} & a_{03} \\
  0 & a_{11} & a_{12} & a_{13} \\
  0 & 0 & a_{22} & a_{23} \\
  0 & 0 & 0 & a_{33}
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  x_1 \\
  x_2 \\
  x_3
\end{bmatrix}
=
\begin{bmatrix}
  b_0 \\
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix}
\]

\[
x_3 = \frac{b_3}{a_{33}}
\]

\[
x_2 = \frac{b_2 - a_{23}x_3}{a_{22}}
\]

\[
x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3}{a_{11}}
\]

\[
x_0 = \frac{b_0 - a_{01}x_1 - a_{02}x_2 - a_{03}x_3}{a_{00}}
\]

*Retro-substitution*
Solving a TRSM Problem using GEMM

\[ AX = \alpha B \]
\[ X = \alpha A^{-1} B \]

• Two operations: **matrix inversion** and **GEMM**
• Matrix inversion has complexity in the order of solving a linear system
• Basically solving the problem twice!
Solving a TRSM Problem using GEMM

\[
\begin{align*}
A_{00}X_0 + A_{01}X_1 + A_{02}X_2 &= \alpha B_0 \\
A_{11}X_1 + A_{12}X_2 &= \alpha B_1 \\
A_{22}X_2 &= \alpha B_2
\end{align*}
\]

- Small blocks leads to fast inversion
- Each block can be inverted in parallel
- Can leverage existing optimized GEMM
- Data stays on the device memory
Solving a TRSM Problem using GEMM

Diagonal Blocks Inversion

Data Locality Pattern on GPU

1. Load from global memory
2. In-place inversion
3. Write to global memory

Global Memory

Input Matrix A

Inverse of the Diagonal blocks of A

Workgroup Local Memory

Work Group 1

Work Group 2

Element being calculated
Elements being accessed
Elements that are not accessed
SYCL Kernel Execution

Runtime graph dependency

Runtime automated kernel scheduling
GEMM-Based TRSM

• Matrix Multiplication

Matrix Multiplication

GEMM - Based TRSM
Performance Evaluation

• Hardware
  • Intel GPU UHD 630
  • ARM Mali G71
  • AMD Radeon RX460

• Libraries
  • SYCL-BLAS
  • clBLAS
  • clBlast
Kernel breakdown for different block sizes
Matrix sizes: 64, 128, 256, 512, 1024, 2048, 4096

AMD Radeon RX 460

ARM Mali G71

Intel GPU UHD 630
Conclusion

• Parametrizable TRSM implementation in SYCL-BLAS
• Step towards performance portability of BLAS operations
  • key component in modern HPC and embedded environments.
• Competitive performance against optimized, vendor-specified libraries
  • clBLAS and clBlast
Future Work

• Diagonal blocks of arbitrary size
• Vectorization of Block Inversion
• No local memory version for devices like the ARM Mali
• Use batched GEMM to accelerate the solver
• Evaluate performance on CPU devices
Thank you!

@codeplaysoft  info@codeplay.com  codeplay.com