High-Performance Sparse Linear Algebra on NVIDIA GPUs with cuSPARSE

Federico Busato | Senior Software Engineer
SIAM CSE21 | March 1, 2021
AGENDA

Background
The GPU Architecture

The cuSPARSE library
Overview, Generic APIs, New Functionalities, cuSPARSELt

Experimental Results
Comparison with Intel MKL and GraphBLAST

cuSPARSE and GraphBLAS
Challenges and Future Directions
THE GPU ARCHITECTURE
A Massively Parallel Processor

- ~220,000 concurrent threads
- 19.5 TFLOPS FMA, 624 TFLOPS Tensor Core
- 2 TB/s Bandwidth
- 40 MB L2 Cache
- 80 GB HBM2e Memory
- Simple programming model and robust ecosystem (compiler, profilers, sanitizer, libraries, etc.)
- Widely adopted: 6M CUDA toolkit downloads every year, 2M registered developers, 69% of Top500 systems
**THE cuSPARSE LIBRARY**

A High-Performance Sparse Linear Algebra Library for Nvidia GPUs

- Part of the CUDA Toolkit since 2010
- APIs and functionalities initially inspired by the Sparse BLAS Standard
  - CSR and COO formats
  - L1 - *Vector-Vector operations*: Axpy, Dot, Rot, Scatter, Gather
  - L2 - *Matrix-Vector operations*: SpMV, Triangular Solver Vector
  - *A few extensions*: SpGEMM, SpGEAM, Conversion operations, preconditioners (incomplete LU/Cholesky, tridiagonal/pentadiagonal solver)
THE cuSPARSE LIBRARY

New Generic APIs

- **Big leap in flexibility:**
  - Mixed-precision computation (e.g. fp16 in/out, fp32 compute)
  - Indexing and Index size: zero-base/one-base indexing and 16-bit, 32-bit, 64-bit sizes
  - Algorithms: deterministic, non-deterministic, memory usage
  - Batched computation, e.g. SpMM: Sigle/Multiple Sparse - Single/Multiple Dense
  - Sparse matrix formats and dense layouts: CSR, CSC, COO, COO AoS + others, row/column-major

- **Inspired by C++ Object Oriented paradigm:** constructor captures all the resources and release them in the destructor

- **Correctness by composability:** each API is responsible for ensuring its properties, e.g. matrix shape, data types, and pointers are checked during the creation

- **Transparent memory management:** no internal allocation

- **Expressive errors:** provide a clear message to understand the problem, do not only rely on error codes

- **Public GitHub examples for each API**

\[
C = \alpha op(A) \cdot op(B) + \beta C
\]
THE cuSPARSE LIBRARY

New Functionalities

Tensor Core Accelerated Block-Sparse Matrix Multiplication

Sample Dense-Dense Matrix Multiplication (SDDMM)
THE cuSPARSeLT LIBRARY
A Specialized CUDA Library for Sparse Matrix - Dense Matrix Multiplication

• Exploit NVIDIA Ampere Architecture Sparse Tensor Core (2:4 sparsity)
  ▶ 624 TFLOPS (31x vs. FMA)
  ▶ 2x vs. dense

• Mixed-precision computation:
  ▶ FP16 inputs/output, FP32 Tensor Core accumulate
  ▶ BFLOAT16 inputs/output, BFLOAT32 Tensor Core accumulate
  ▶ INT8 inputs/output, INT32 Tensor Core compute

• Future releases will likely add support for activation functions, e.g. ReLU(\(\alpha \cdot op(A)op(B) + \beta op(C) + bias\)), and TensorFloat-32
**PERFORMANCE COMPARISON**

cuSPARSE vs. Intel MKL

**CSR SpMV**

- Intel MKL
- NVIDIA cuSPARSE V100
- NVIDIA cuSPARSE A100

**CSR SpMM**

- Intel MKL
- NVIDIA cuSPARSE V100
- NVIDIA cuSPARSE A100

Benchmark results and performance comparisons are shown for different workloads and hardware configurations.
PERFORMANCE COMPARISON

cuSPARSE vs. GraphBLAST

- GPU implementation of Sparse Matrix - Dense Matrix Multiplication (SpMM) is not currently available in GraphBLAST
- Sparse Matrix - Sparse Matrix Multiplication (SpGEMM) uses cuSPARSE old APIs. New Generic API provides ~5x performance improvement
**cuSPARSE and GraphBLAS**

Challenges and Future Directions

*cuSPARSE* is a sparse linear algebra library. *GraphBLAS* does not strictly rely on standard linear algebra but on its small extensions...

- Semiring computation (operators), Masking

...it not so different from *deep learning*

- Activation functions, on-the-fly network pruning

**Challenges and future directions:**

- Make *generic* a closed-source device library to support arbitrary operators
- Better support for *Sparse Matrix - Sparse Vector* (SpMSpV)
- Add support for matrix *masking*
THANK YOU

WE ARE **HIRING** AND OPEN FOR **COLLABORATIONS**

contact fbusato@nvidia.com