AGENDA

CUTLASS 1.2

cuBlasLt and Matmul

Reduced Precision and Reproducibility
AGENDA

CUTLASS 1.2
CUTLASS MOTIVATION

Productivity Challenges in DLA and Deep Learning

Problem:

Multiplicity of Algorithms and Data Types
- GEMM, Convolution, Back propagation
- Mixed precision arithmetic

Kernels specialized for layout and problem size
- NT, TN, NCHW, NHWC

Kernel Fusion
- Custom operations composed with GEMM and convolution

Solution:

Template Library for Linear Algebra Computations in CUDA C++
- https://github.com/NVIDIA/cutlass
- CUTLASS Parallel for All blog post,
- GTC 2018 CUTLASS talk [video recording]

Data movement and computation primitives
- Iterators, matrix fragments, matrix computations

Inspired by CUB

Merrill, Duane. NVIDIA CUB. http://nvlabs.github.io/cub/
GENERIC PROGRAMMING FOR DLA AND DEEP LEARNING

Write code once to exploit common design patterns

- Matrix multiply: grid-, CTA-, warp-, thread-level scope
- Tile: bounded region of pitch-linear memory
- Tile Iterator: efficient load, store, and traversal over a sequence of tiles
- Fragment: partition tile elements among threads

CUDA C++ as a *declarative* programming language
## IMPLEMENTED COMPUTATIONS

### CUTLASS v1.2

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGEMM</td>
<td>float</td>
<td>float</td>
<td>float</td>
</tr>
<tr>
<td></td>
<td>half</td>
<td>half</td>
<td>half, float</td>
</tr>
<tr>
<td>DGEMM</td>
<td>double</td>
<td>double</td>
<td>double</td>
</tr>
<tr>
<td>HGEMM</td>
<td>half</td>
<td>half</td>
<td>half</td>
</tr>
<tr>
<td>IGEMM</td>
<td>int8_t</td>
<td>int8_t</td>
<td>int32_t</td>
</tr>
<tr>
<td></td>
<td>int8_t</td>
<td>int8_t</td>
<td>int32_t</td>
</tr>
<tr>
<td>WMMA GEMM</td>
<td>half</td>
<td>half</td>
<td>half</td>
</tr>
<tr>
<td></td>
<td>half</td>
<td>half</td>
<td>float</td>
</tr>
<tr>
<td></td>
<td>int8_t</td>
<td>int8_t</td>
<td>int32_t</td>
</tr>
<tr>
<td></td>
<td>int4_t</td>
<td>int4_t</td>
<td>int32_t</td>
</tr>
<tr>
<td></td>
<td>bin1_t</td>
<td>bin1_t</td>
<td>int32_t</td>
</tr>
</tbody>
</table>

+ Batched strided; optimizations for small GEMM
CUTLASS 1.2 PROGRESS

What's New in CUTLASS 1.2

October 2018

- Parallelized Reductions
- Batched strided WMMA GEMM

What's New in CUTLASS 1.1

September 2018

- CUTLASS Documentation
- Examples
  - Basic GEMM, tensor views, CUTLASS utilities, batched GEMM, WMMA GEMM
- Turing Features
  - WMMA GEMM targeting TensorCores - INT8, INT4, 1-bit
- Batched Strided GEMM
- Threadblock rasterization strategies
  - Improved performance for adverse problem sizes and data layouts
- Extended CUTLASS Core components
  - Tensor views support arbitrary matrix and tensor layouts
  - Zip iterators for structuring multiple data streams
- Enhanced CUTLASS utilities
  - Reference implementations for tensor operations in host and device code
  - Added HostMatrix<> for simplified matrix creation
CUTLASS Performance Relative to cuBLAS
Titan V - CUDA 10.0
COMPLETE GEMM STRUCTURAL MODEL

Embodied by CUTLASS CUDA templates

Global Load Stream

Shared Load Stream

Matrix Multiply

Epilogue

GemmGlobalIteratorAb

Transformer

GemmSharedStoreTileAb

GemmSharedLoadTile\{A,B\}

fma, dp4a, hfma2

WMMA

Transformer

GemmSharedStoreTileD

GemmSharedLoadTileD

GemmGlobalIteratorC

Functior

GemmGlobalIteratorD
AGENDA

cuBLASLt and Matmul
CUBLASLT
The GEMM Library

Today

- SASS Kernels
- CUTLASS (IMMA, FP16)
- CUDA Kernels

Future

- cuBLAS_Legacy
  - cuBLAS_Legacy Context
    - BLAS 1, 2, 3 (subset)
- CUDA Kernels

- cuBLASLt
  - cuBLASLt Context
    - Matmul
      - SASS Kernels
      - CUTLASS

cublasStatus_t

cublasLtMatmul(cublasLtHandle_t handle,
               cublasLtMatmulDesc_t computeDesc,
               const void *alpha, /* host or device pointer */
               const void *A,
               cublasLtMatrixLayout_t Adesc,
               const void *B,
               cublasLtMatrixLayout_t Bdesc,
               const void *beta, /* host or device pointer */
               const void *C,
               cublasLtMatrixLayout_t Cdesc,
               void *D,
               cublasLtMatrixLayout_t Ddesc,
               const cublasLtMatmulAlgo_t *algo,
               void *workSpace,
               size_t workSpaceSizeInBytes,
               cudaStream_t stream) {

MATMUL
Omnibus GEMM
MATMUL ALGO

Flexible heuristics and implementations

• Parameters programmability

• Adaptive algorithm heuristics
  • Construct plans like FFTW_ESTIMATE/FFTW_MEASURE/FFTW_EXHAUSTIVE
  • Split-K, Reduction Algs
  • MMA (Tensor cores)
  • Logical reordering of compute elements (Multi-GPU, Cache-Adaptive, etc)
  • Math-mode (Atomics, Repro, HiPrec, Gaussian, etc)
AGENDA

Reduced Precision and Reproducibility
REDUCED PRECISION
GemmEx and Matmul

- `cublasGemmEx()`
- `cublasGemmBatchedEx()`
- `cublasGemmStridedBatchedEx()`
- `cublasLtMatmul()`

<table>
<thead>
<tr>
<th>Compute type</th>
<th>A/B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA_R_16F</td>
<td>CUDA_R_16F</td>
<td>CUDA_R_16F</td>
</tr>
<tr>
<td>CUDA_R_32F</td>
<td>CUDA_R_16F</td>
<td>CUDA_R_16F</td>
</tr>
<tr>
<td></td>
<td>CUDA_R_32F</td>
<td>CUDA_R_32F</td>
</tr>
<tr>
<td></td>
<td>CUDA_R_8I</td>
<td>CUDA_R_32F</td>
</tr>
<tr>
<td></td>
<td>CUDA_R_32F</td>
<td>CUDA_R_32F</td>
</tr>
<tr>
<td>CUDA_R_64F</td>
<td>CUDA_R_64F</td>
<td>CUDA_R_64F</td>
</tr>
<tr>
<td>CUDA_C_32F</td>
<td>CUDA_C_8I</td>
<td>CUDA_C_32F</td>
</tr>
<tr>
<td></td>
<td>CUDA_C_32F</td>
<td>CUDA_C_32F</td>
</tr>
<tr>
<td></td>
<td>CUDA_C_64F</td>
<td>CUDA_C_64F</td>
</tr>
</tbody>
</table>
NOTE ON REPRODUCIBILITY

By design, all cuBLAS routines with
- Same toolkit
- Same architecture (same #SMs)
generate the same bit-wise results every run.

For some routines like
- `cublas<T>symv`
- `cublas<T>hemv`
reproducibility can be sacrificed for efficiency with
- `cublasSetAtomicsMode()`