BLAS AT NVIDIA

Cris Cecka

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

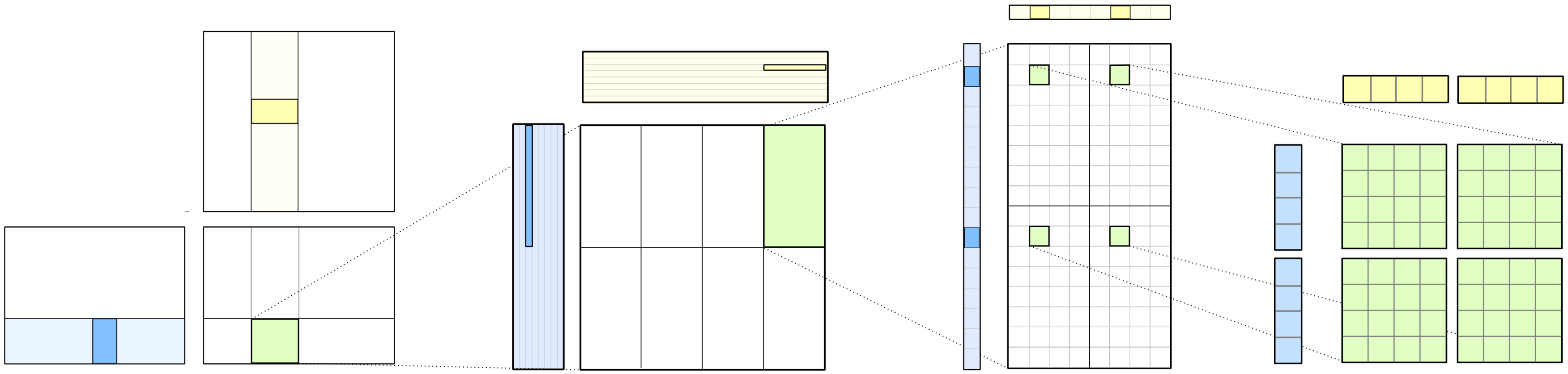
Inspired by CUB

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



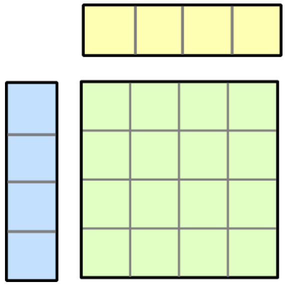
CUDA C++ as a *declarative* programming language

IMPLEMENTED COMPUTATIONS

CUTLASS v1.2

	A	B	C	Accumulator
SGEMM	float	float	float	float
	half	half	half, float	float
DGEMM	double	double	double	double
HGEMM	half	half	half	half
IGEMM	int8_t	int8_t	int8_t	int32_t
	int8_t	int8_t	float	int32_t
WMMA GEMM	half	half	half	half
	half	half	half, float	float
	int8_t	**int8_t**	**int32_t**	**int32_t**
	int4_t	**int4_t**	**int32_t**	**int32_t**
	bin1_t	**bin1_t**	**int32_t**	**int32_t**

+ Batched strided; optimizations for small GEMM



CUTLASS 1.2 PROGRESS

What's New in CUTLASS 1.2

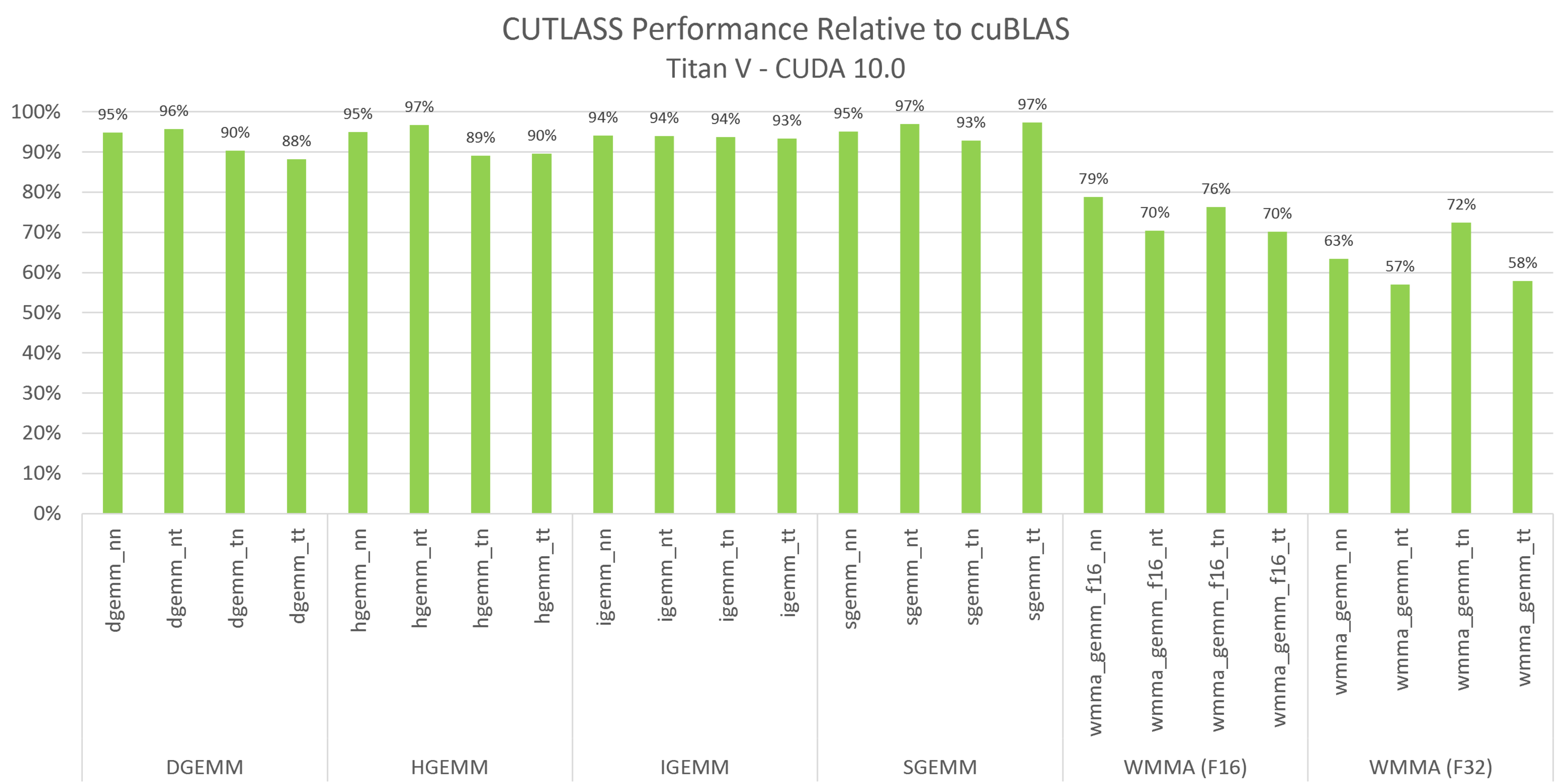
October 2018

- Parallelized Reductions
- Batched strided WMMA GEMM

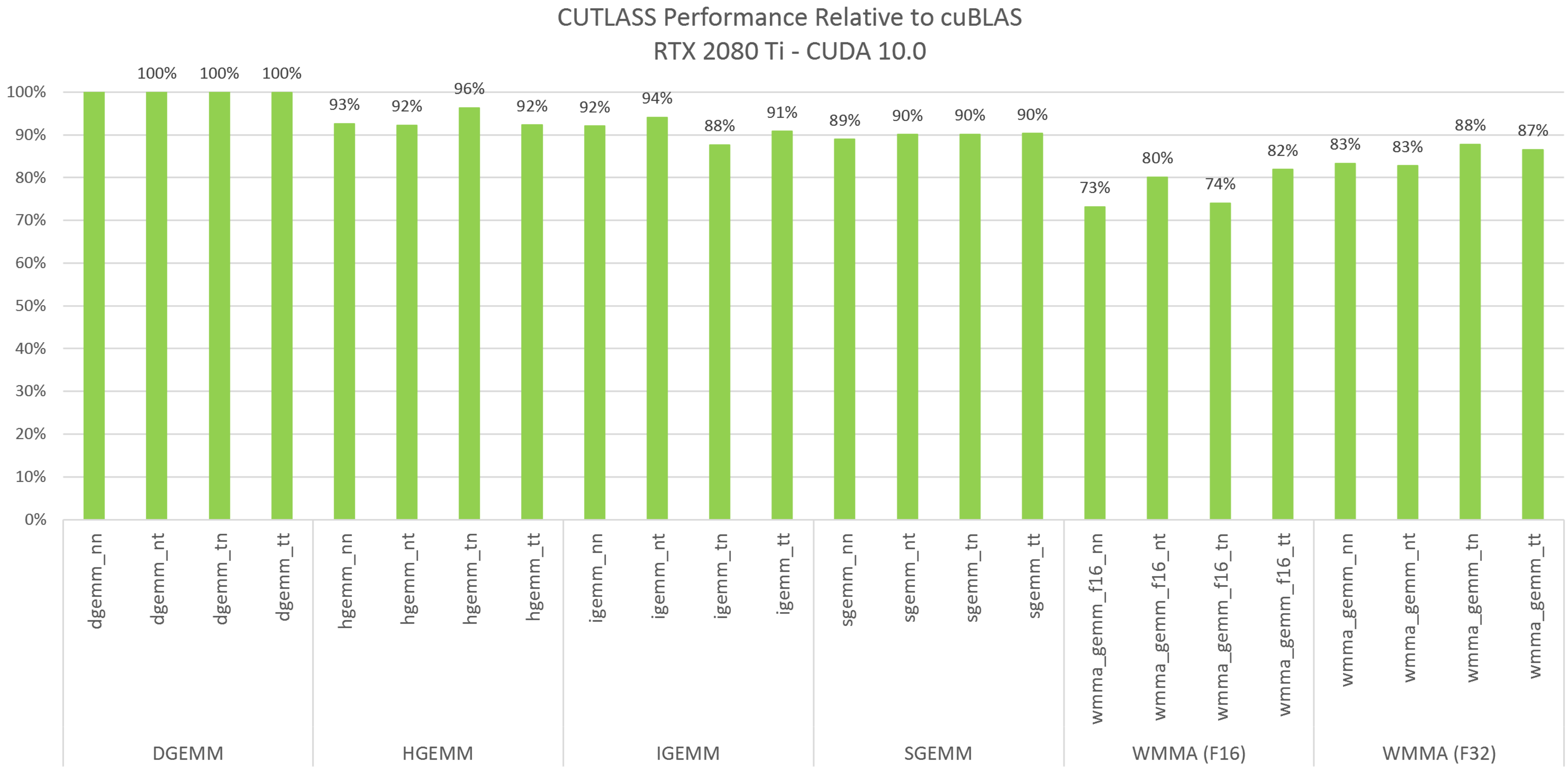
What's New in CUTLASS 1.1

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

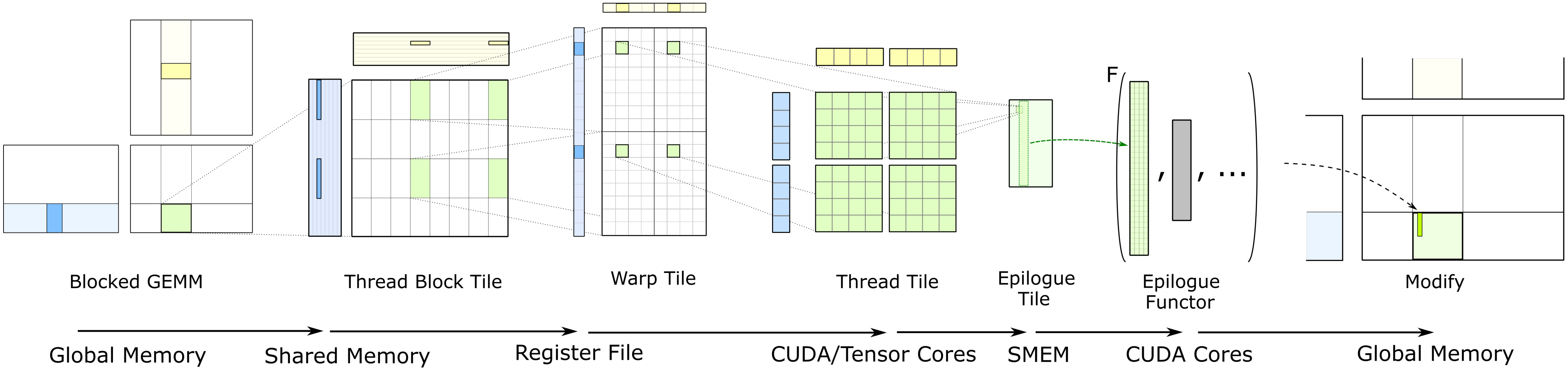


CUTLASS Performance Relative to cuBLAS
Titan V - CUDA 10.0
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
95%
96%
90%
88%
95%
97%
89%
90%
94%
94%
94%
93%
95%
97%
93%
97%
79%
70%
76%
70%
63%
57%
72%
58%
dgemm_nn
dgemm_nt
dgemm_tn
dgemm_tt
hgemm_nn
hgemm_nt
hgemm_tn
hgemm_tt
igemm_nn
igemm_nt
igemm_tn
igemm_tt
sgemm_nn
sgemm_nt
sgemm_tn
sgemm_tt
wmma_gemm_f16_nn
wmma_gemm_f16_nt
wmma_gemm_f16_tn
wmma_gemm_f16_tt
wmma_gemm_nn
wmma_gemm_nt
wmma_gemm_tn
wmma_gemm_tt
DGEMM
HGEMM
IGEMM
SGEMM
WMMA (F16)
WMMA (F32)


CUTLASS Performance Relative to cuBLAS
RTX 2080 Ti - CUDA 10.0
100%
100%
100%
93%
92%
96%
92%
92%
94%
88%
91%
89%
90%
90%
90%
73%
80%
74%
82%
83%
83%
88%
87%
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
dgemm_nn
dgemm_nt
dgemm_tn
dgemm_tt
hgemm_nn
hgemm_nt
hgemm_tn
hgemm_tt
igemm_nn
igemm_nt
igemm_tn
igemm_tt
sgemm_nn
sgemm_nt
sgemm_tn
sgemm_tt
wmma_gemm_f16_nn
wmma_gemm_f16_nt
wmma_gemm_f16_tn
wmma_gemm_f16_tt
wmma_gemm_nn
wmma_gemm_nt
wmma_gemm_tn
wmma_gemm_tt
DGEMM
HGEMM
IGEMM
SGEMM
WMMA (F16)
WMMA (F32)

COMPLETE GEMM STRUCTURAL MODEL

Embodied by CUTLASS CUDA templates



GemmGlobalIteratorAb
Transformer
GemmSharedStoreTileAb

GemmSharedLoadTile{A,B}

fma, dp4a, hfma2
WMMA

Transformer
GemmSharedStoreTileD
GemmSharedLoadTileD

GemmGlobalIteratorC
Functor
GemmGlobalIteratorD

Global Load Stream

Shared Load Stream

Matrix Multiply

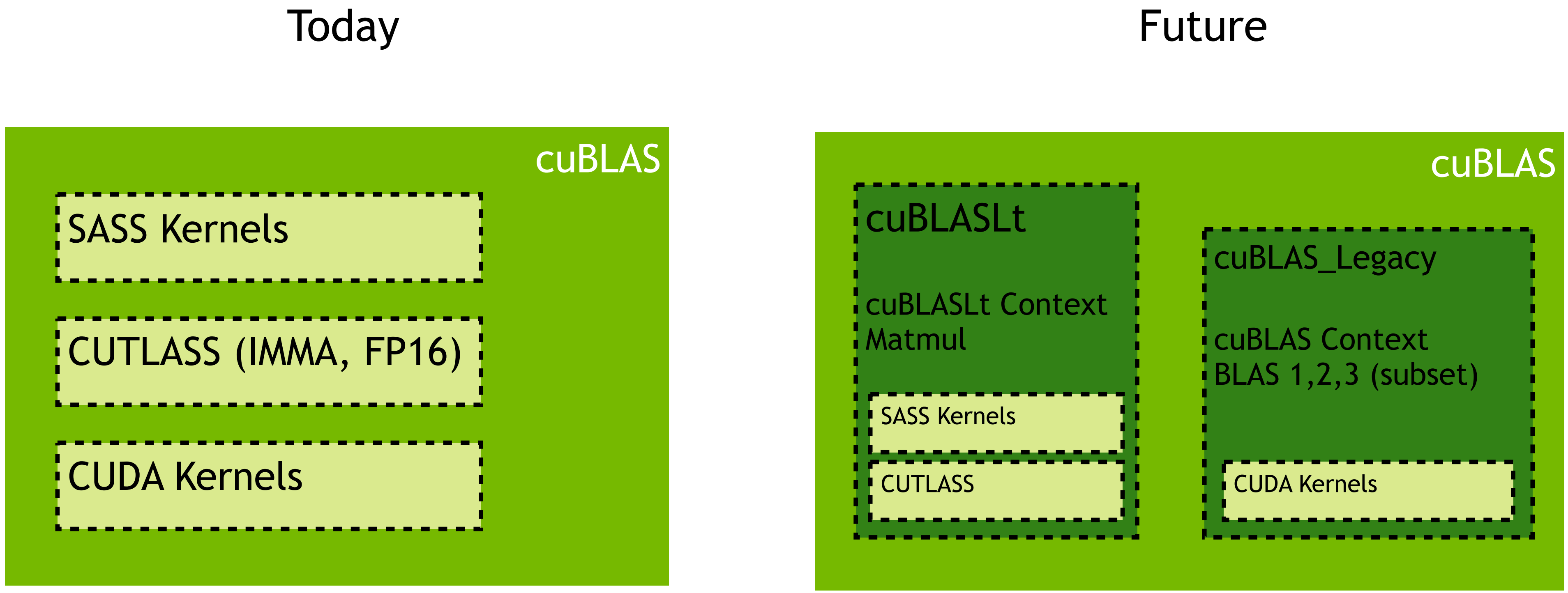
Epilogue

AGENDA

cuBLASLt and Matmul

CUBLASLT

The GEMM Library



MATMUL

Omnibus GEMM

```
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) {
```

```
cublasLtMatrixLayout_t

cublasLtMatrixLayoutCreate(…)
cublasLtMatrixLayoutDestroy(…)
cublasLtMatrixLayoutSetAttribute(…)
cublasLtMatrixLayoutGetAttribute(…)
```

```
cublasLtMatmulDesc_t

cublasLtMatmulDescCreate(…)
cublasLtMatmulDescDestroy(…)
cublasLtMatmulDescSetAttribute(…)
cublasLtMatmulDescGetAttribute(…)
```

```
cublasLtMatmulAlgo_t

cublasLtMatmulAlgoInit(…)
cublasLtMatmulAlgoFind(…)
cublasLtMatmulAlgoGetHeuristic(…)
cublasLtMatmulAlgoConfigSetAttribute(…)
cublasLtMatmulAlgoConfigGetAttribute(…)
```

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()`

Compute type	A/B	C
CUDA_R_16F	CUDA_R_16F	CUDA_R_16F
CUDA_R_32F	CUDA_R_16F	CUDA_R_16F
	CUDA_R_16F	CUDA_R_32F
	CUDA_R_8I	CUDA_R_32F
	CUDA_R_32F	CUDA_R_32F
CUDA_R_64F	CUDA_R_64F	CUDA_R_64F
CUDA_C_32F	CUDA_C_8I	CUDA_C_32F
	CUDA_C_32F	CUDA_C_32F
CUDA_C_64F	CUDA_C_64F	CUDA_C_64F

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()`

NVIDIA