GETTING STARTED WITH TENSOR CORES IN HPC

Vishal Mehta, NVIDIA

Super Computing, 2019

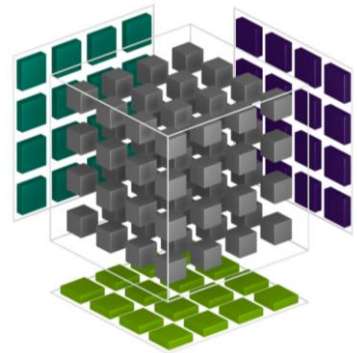
AGENDA

- Tensor Cores Architecture
- Programming Approaches
  - DL Framework
  - Libraries
  - WMMA CUDA
  - CUTLASS
- HPC Case Studies
  - Particle in Cell
  - Spherical Harmonics - IFS

VOLTA ARCHITECTURE AND TENSOR CORES

TENSOR CORE

Mixed Precision Matrix Math
4x4 matrices

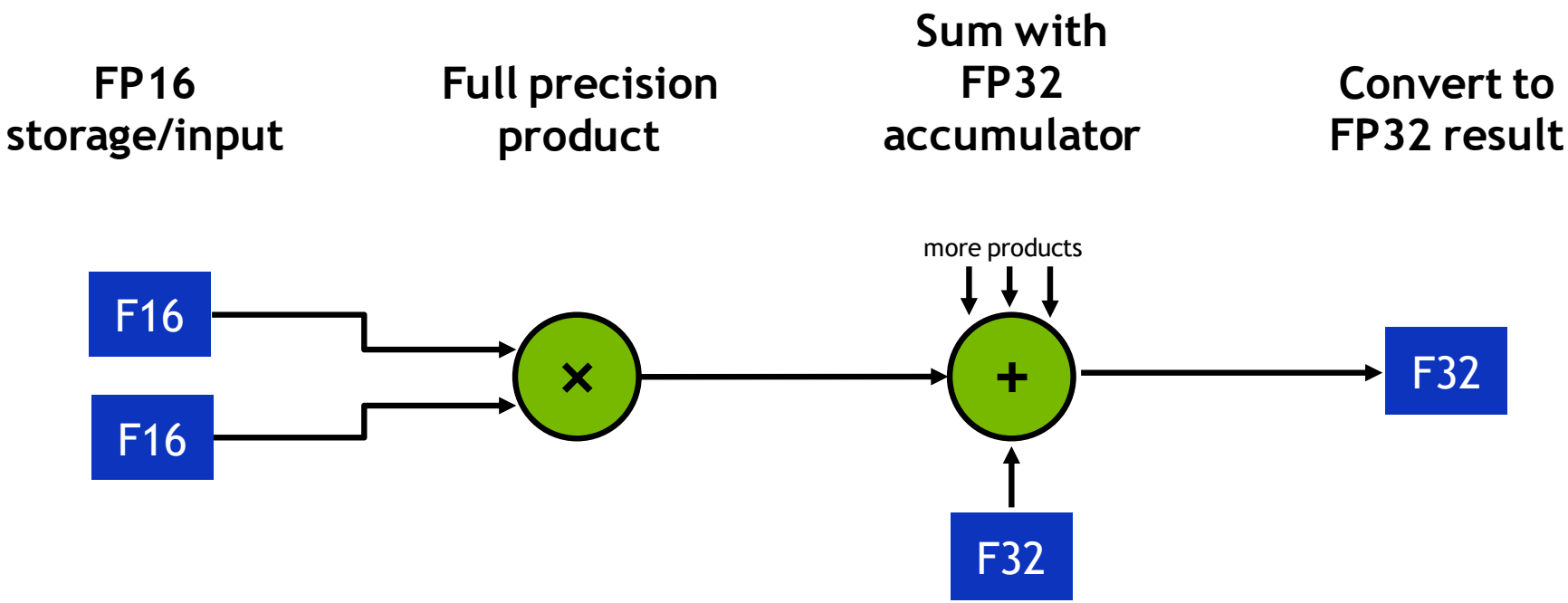


$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32 FP16 FP16 FP16 or FP32

$$D = AB + C$$

VOLTA TENSOR OPERATION



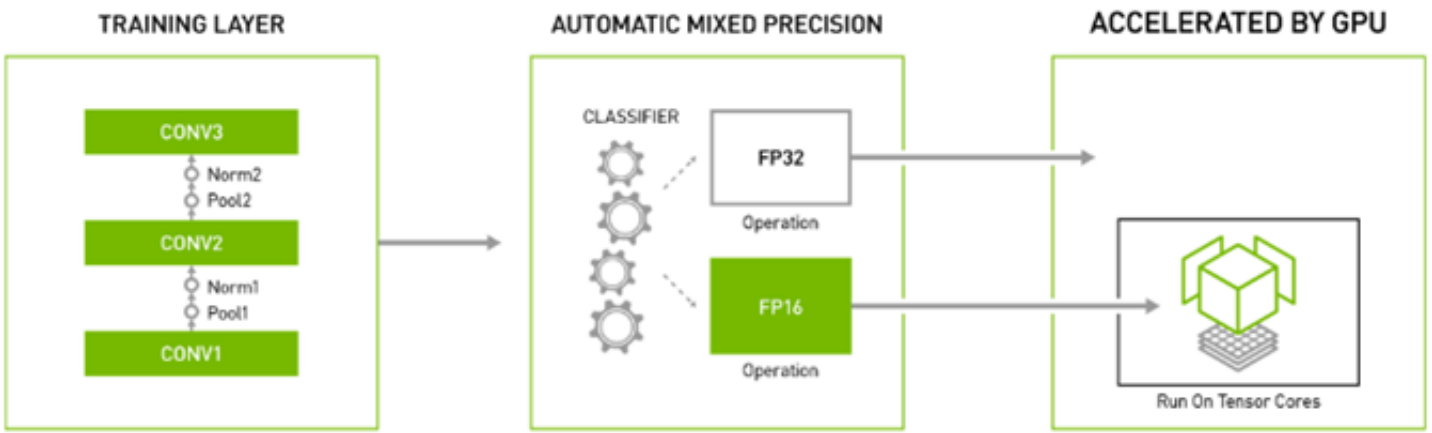
Also supports FP16 accumulator mode for inferencing

TENSOR CORE PROGRAMMING MODELS

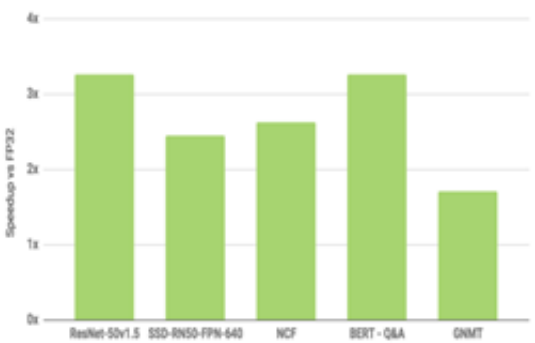
DEEP LEARNING FRAMEWORKS

AUTOMATIC MIXED PRECISION

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```



TensorFlow
Upto 3X Speedup



Insert ~ two lines of code to introduce Automatic Mixed-Precision and get upto 3X speedup

AMP uses a graph optimization technique to determine FP16 and FP32 operations

Support for TensorFlow, PyTorch and MXNet

CUDA LIBRARIES

CUBLAS TENSOR CORE HOW-TO

CUBLAS functions	mathMode = CUBLAS_DEFAULT_MATH	mathMode = CUBLAS_TENSOR_OP_MATH
cublasHgemm, cublasSgemm, cublasGemmEx(algo=DEFAULT)	Disallowed	Allowed
cublasGemmEx(algo=*_TENSOR_OP	Allowed	Allowed

Math Mode set with **cublasSetMathMode** function.

Volta and Turing family Tensor Core can be used with in mixed precision (FP16 inputs, FP32 accumulation, FP16 or FP32 output) routines.

Pure single precision routines use tensor core (when allowed) by down-converting inputs to half (FP16) precision on the fly.

Constraint: M,N,K,LDA,LDB,LDC and A,B,C pointers must ALL be aligned to 8 because of high memory bandwidth needed to efficiently use Tensor Cores.

HGEMM VS GEMMEX

```
cublasSetMathMode(handle, CUBLAS_TENSOR_OP_MATH);
const __half *A = ...;
const __half *B = ...;
      __half *C = ...;
cublasHgemm(handle, transa, transb, m, n, k,
            alpha, A, lda, B, ldb,
            beta, C, ldc);
```

accumulates in FP16

exact results only up to N < 2048

vs.

```
...
      float *C = ...;
cublasGemmEx(handle, transa, transb, m, n, k,
             alpha, A, CUDA_R_16F, lda, B, CUDA_R_16F, ldb,
             beta, C, CUDA_R_32F, ldc,
             CUDA_R_32F,
             CUBLAS_GEMM_DEFAULT_TENSOR_OP);
```

accumulates in FP32

nearly as fast as cublasHgemm
(same datapath, just a bit more I/O)

exact results up to N < 2^{24}

make sure to ask for tensor cores!

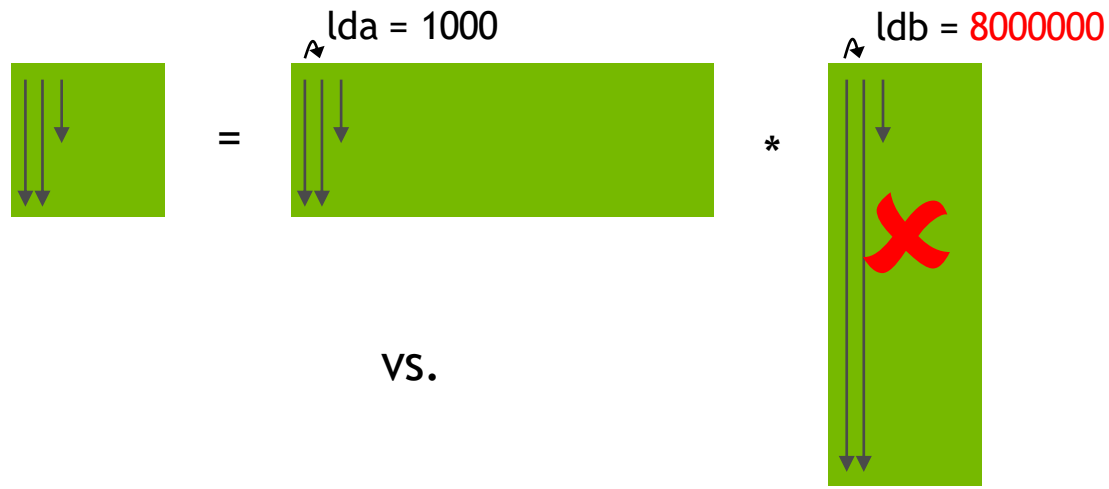
PERFORMANCE TIPS FOR CUBLAS

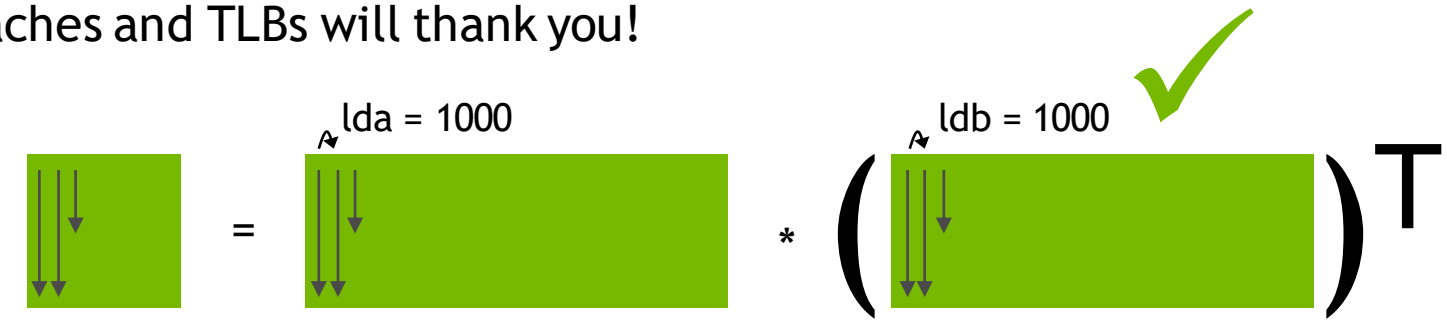
When N gets large, A and B matrices can get very long and skinny

Prefer the memory layout that keeps lda and ldb small

Change the transa/transb parameters on cublas*Gemm* to match

Your caches and TLBs will thank you!





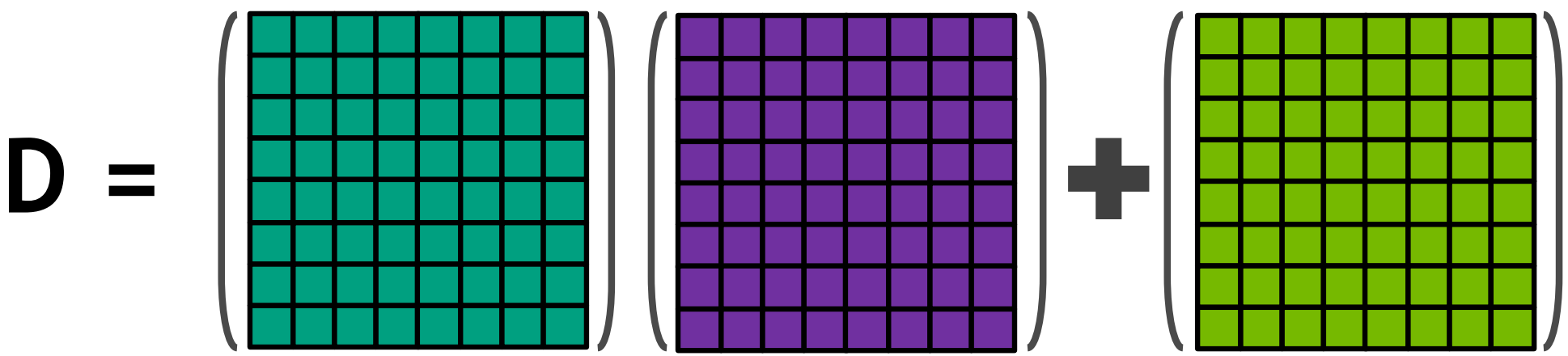
CUDA WMMA

CUDA TENSOR CORE PROGRAMMING

WMMA Matrix Multiply and Accumulate Operation

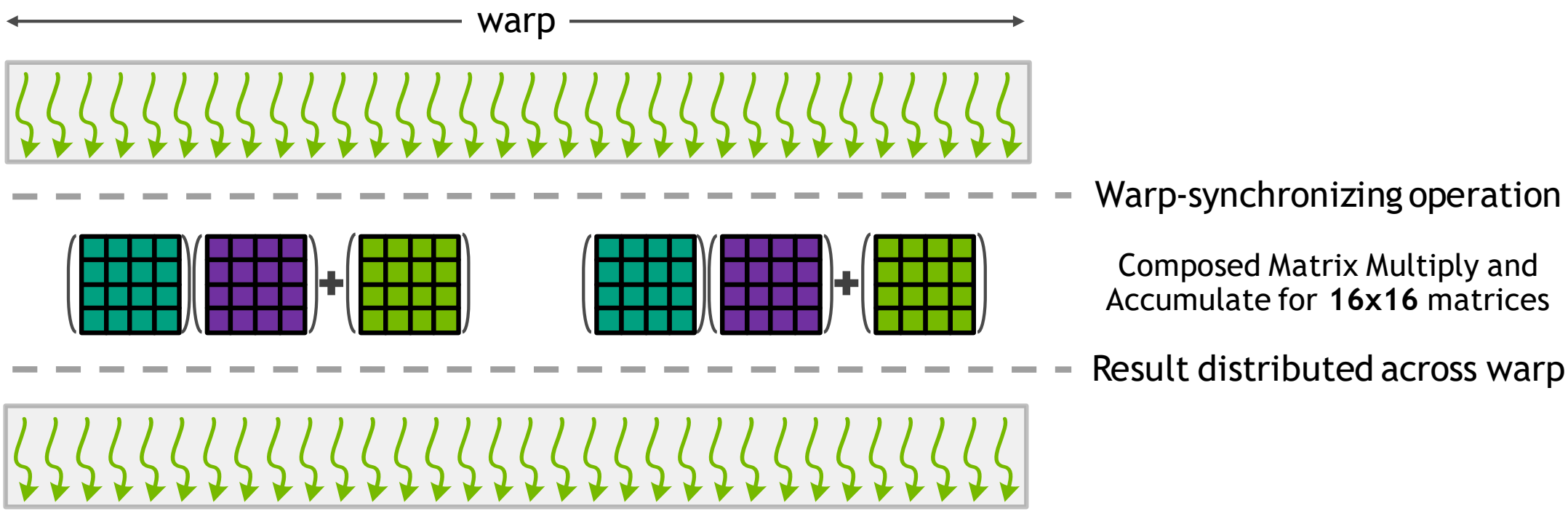
Warp-level operation to perform matrix multiply and accumulate

```
wmma::mma_sync(Dmat, Amat, Bmat, Cmat);
```



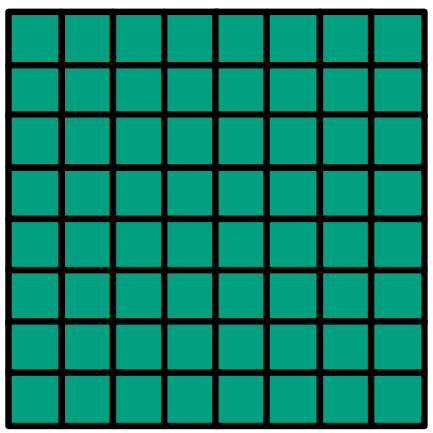
TENSOR SYNCHRONIZATION

Full Warp 16x16 Matrix Math



CUDA TENSOR CORE PROGRAMMING

WMMA datatypes



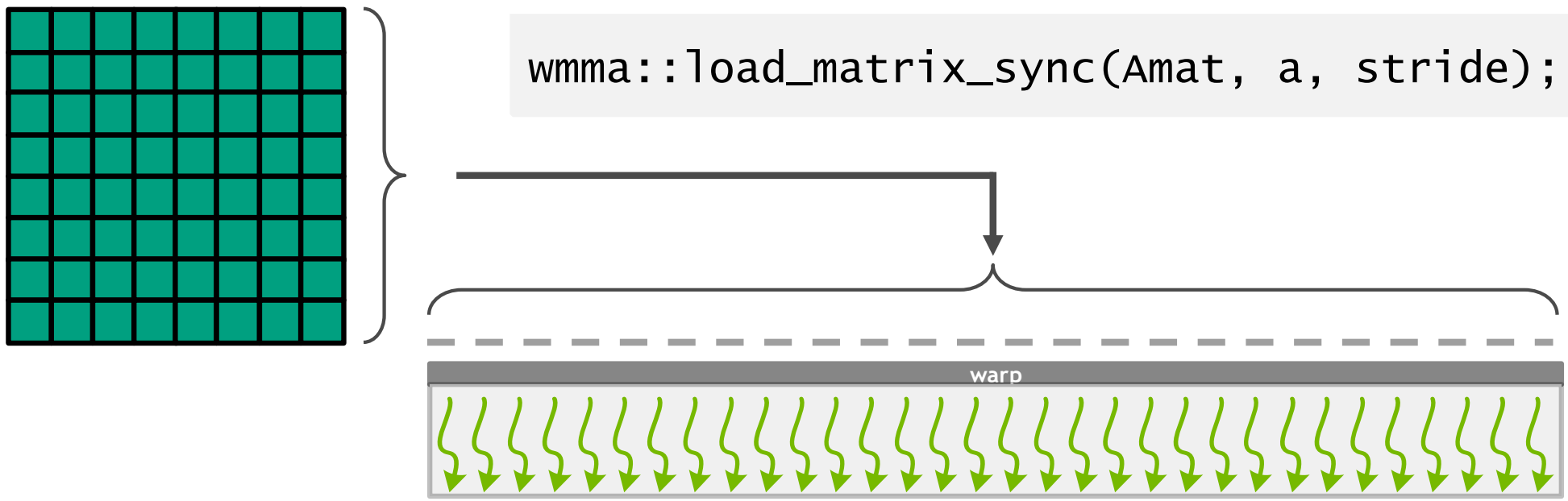
Per-Thread fragments to hold components of matrices for use with Tensor Cores

```
wmma::fragment<matrix_a, …> Amat;
```

CUDA TENSOR CORE PROGRAMMING

WMMA load and store operations

Warp-level operation to fetch components of matrices into fragments



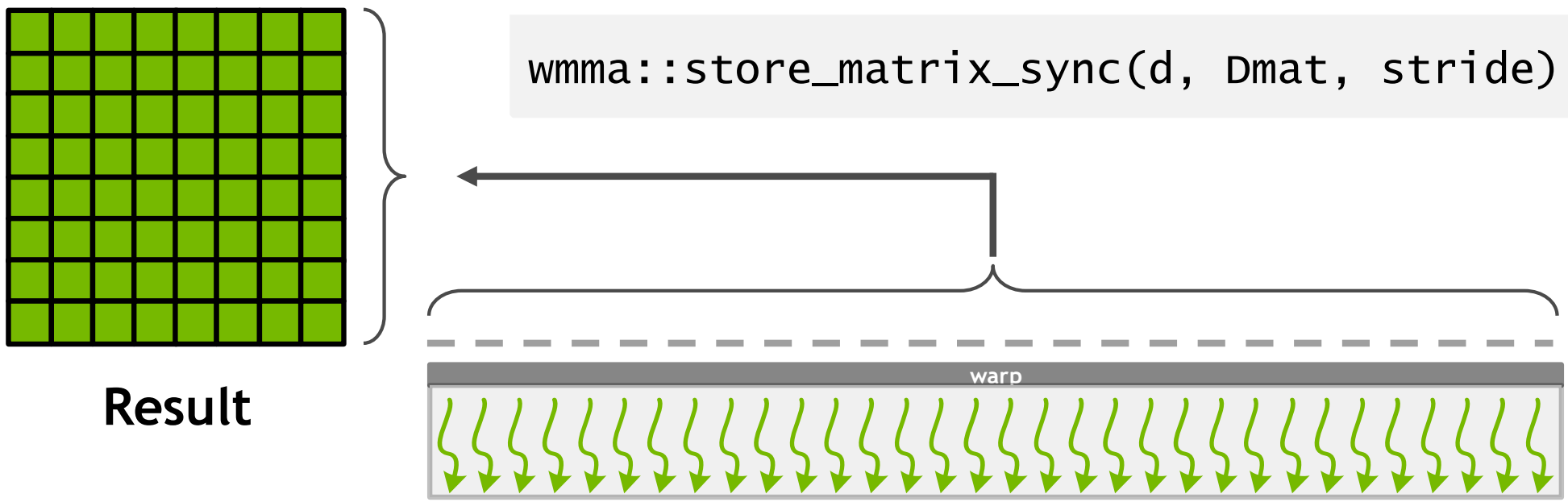

```
wmma::load_matrix_sync(Amat, a, stride);
```

CUDA TENSOR CORE PROGRAMMING

WMMA load and store operations

Warp-level operation to fetch components of matrices into fragments

```
wmma::store_matrix_sync(d, Dmat, stride);
```



TENSOR CORE EXAMPLE

Create Fragments

Initialize Fragments

Perform MatMul

Store Results

```
__device__ void tensor_op_16_16_16(
    float *d, half *a, half *b, float *c)
{
  wmma::fragment<matrix_a, …> Amat;
  wmma::fragment<matrix_b, …> Bmat;
  wmma::fragment<matrix_c, …> Cmat;

  wmma::load_matrix_sync(Amat, a, 16);
  wmma::load_matrix_sync(Bmat, b, 16);
  wmma::fill_fragment(Cmat, 0.0f);

  wmma::mma_sync(Cmat, Amat, Bmat, Cmat);

  wmma::store_matrix_sync(d, Cmat, 16,
                          wmma::row_major);
}
```

CUDA C++

Warp-Level Matrix Operations

TENSOR CORES IN CUDA FORTRAN

Similar to CUDA C WMMA API, with some name changes

real(2) support for half-precision data available (on both host and device) in PGI 19.7 compilers

Requires **`wmma`** Fortran module and macros in **`cuf_macros.CUF`** file

CUDA FORTRAN TENSOR CORE EXAMPLE

Device Code

WMMA Definitions

WMMA "fragments"

Assignment overloaded to call fill_fragment()

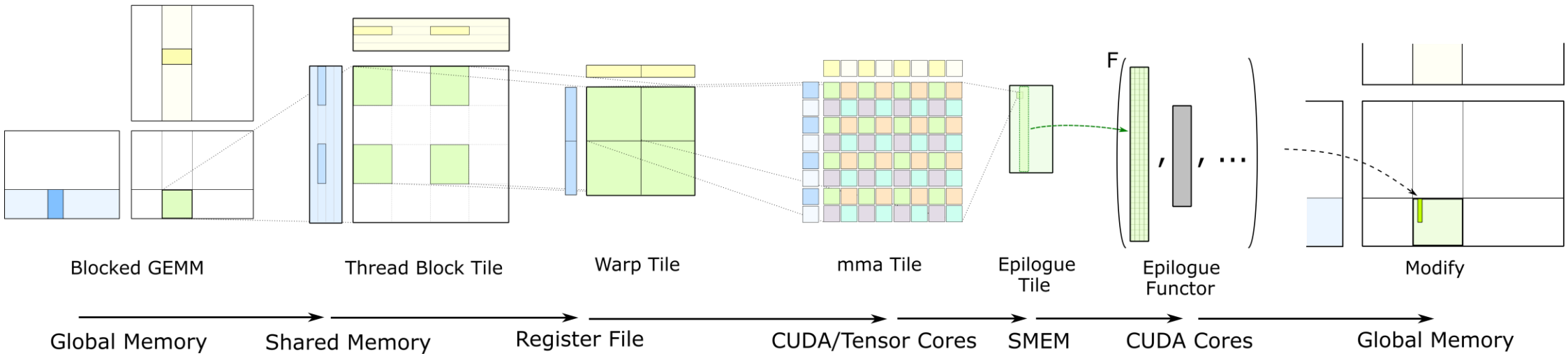
```
#include "cuf_macros.CUF"
module m
contains
  attributes(global) subroutine wmma_16x16(a, b, c)
    use wmma
    real(2), intent(in) :: a(16,*), b(16,*)
    real(4) :: c(16,*)
    WMMASubMatrix(WMMAMatrixA, 16, 16, 16, Real, WMMAColMajor) :: sa
    WMMASubMatrix(WMMAMatrixB, 16, 16, 16, Real, WMMAColMajor) :: sb
    WMMASubMatrix(WMMAMatrixC, 16, 16, 16, Real, WMMAKind4) :: sc

    sc = 0.0_4
    call wmmaLoadMatrix(sa, a(1,1), 16)
    call wmmaLoadMatrix(sb, b(1,1), 16)
    call wmmaMatMul(sc, sa, sb, sc)
    call wmmaStoreMatrix(c(1,1), sc, 16)
  end subroutine wmma_16x16
end module m
```

CUTLASS

CUTLASS 1.3

CUDA C++ Template Library for Matrix Algebra



CUTLASS template library for GEMM computations

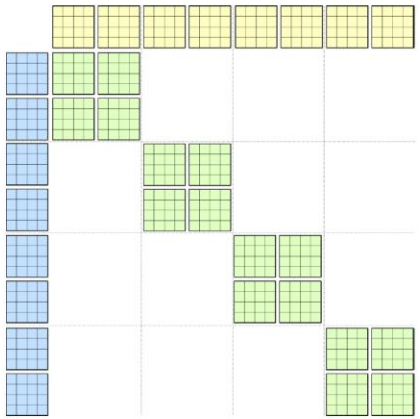
- Blocked structure to maximize data reuse
- Software pipelined to hide latency
- Conflict-free Shared Memory access to maximize data throughput

See CUTLASS GTC 2018 talk.

TENSOR CORES

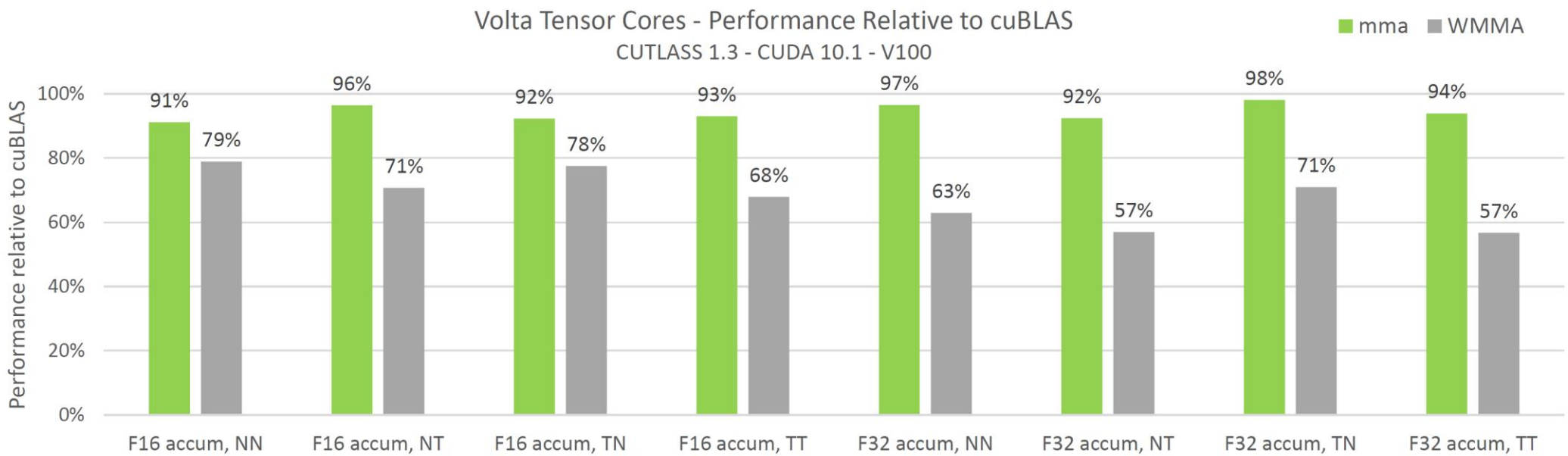
Tensor Cores

- **8x speedup** for mixed-precision matrix multiply
- Programmable via WMMA API (CUDA 9)



Direct access to Volta Tensor Cores: **mma.sync** (new instruction in CUDA 10.1)

- Maximum efficiency on Volta SM Architecture
- New in CUTLASS 1.3



https://on-demand-gtc.gputechconf.com/gtcnew/sessionview.php?sessionName=s9593-cutensor%3a+high-performance+tensor+operations+in+cuda

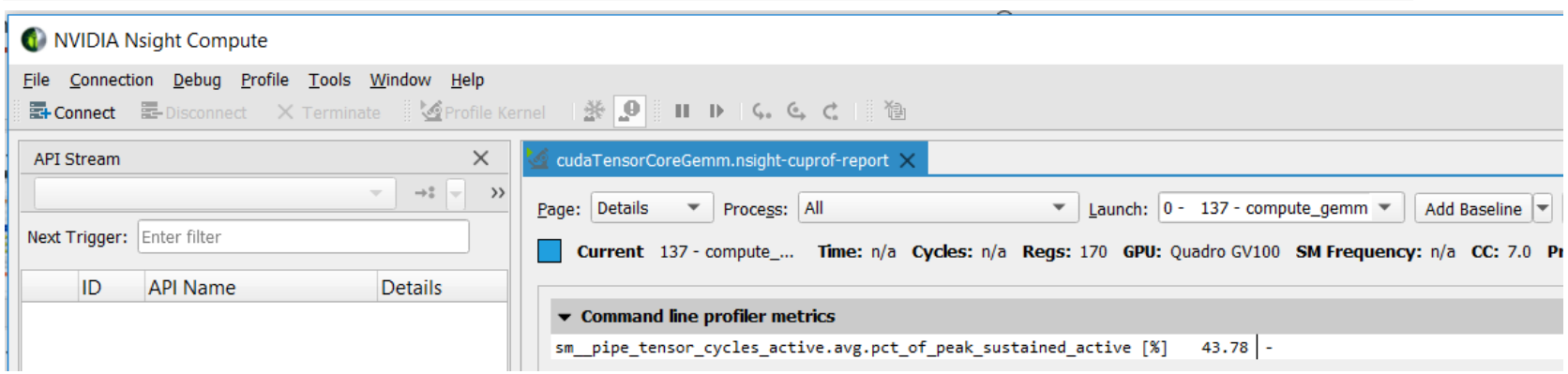
PROFILING

TENSOR CORES WITH NSIGHT COMPUTE

- The Nsight Compute CLI allows collecting several metrics related to tensor core usage
- This data can be view from the CLI or via the Nsight Compute GUI

```
nv-nsight-cu-cli --metrics sm__pipe_tensor_cycles_active.avg.pct_of_peak_sustained_active ./cudaTensorCoreGemm

compute_gemm, 2019-Aug-08 12:48:39, Context 1, Stream 7
  Section: Command line profiler metrics
  ---------------------------------------------------------------------- --------------- ------------------------------
  sm__pipe_tensor_cycles_active.avg.pct_of_peak_sustained_active                    %                          43.44
  ---------------------------------------------------------------------- --------------- ------------------------------
```



HPC CASE STUDIES

PARTICLE PUSH IN MAGNETIC FIELD USING TENSOR CORE (PARTICLE IN CELL)

PARTICLE PUSH

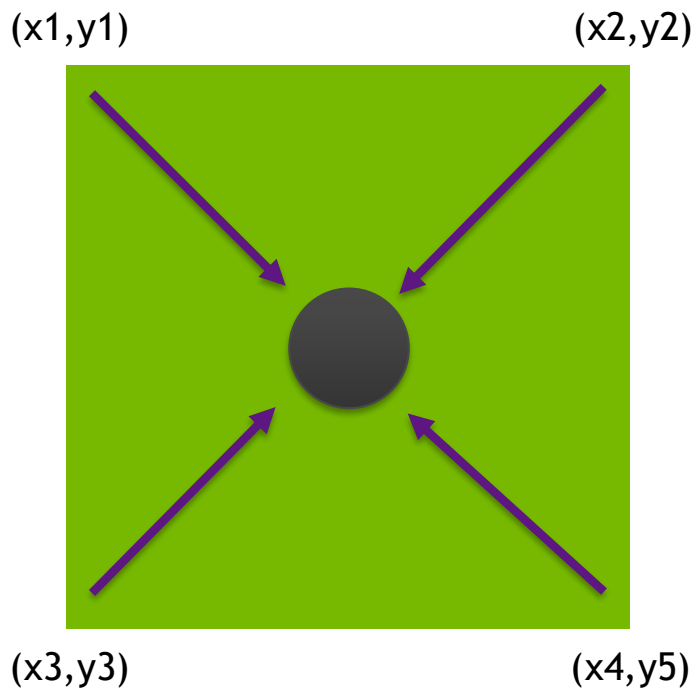
- The governing equation for particle velocity in magnetic field is given by:
- $\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{q}{m(\mathrm{v \times B})}, v = velocity, q = charge, m = mass, B = magnetic\ field$
- *Discretizing the above equation in 2 dimension can lead to :

```
/*grab magnetic field at current position*/
B=EvalB(x);

/*get new velocity at n+1*/
v2[0] = v[0] + q/m*B*v[1]*dt;
v2[1] = v[1] - q/m*B*v[0]*dt;

/*update position*/
x2[0] = x[0] + v2[0]*dt;
x2[1] = x[1] + v2[1]*dt;

/*push down*/
v[0]=v2[0];
v[1]=v2[1];
```

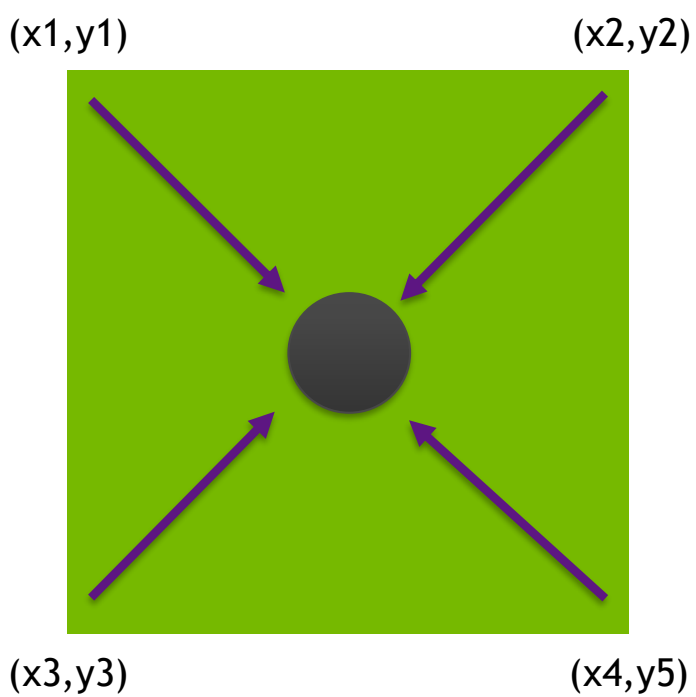


Gather by magnetic forces from the cell vertices.

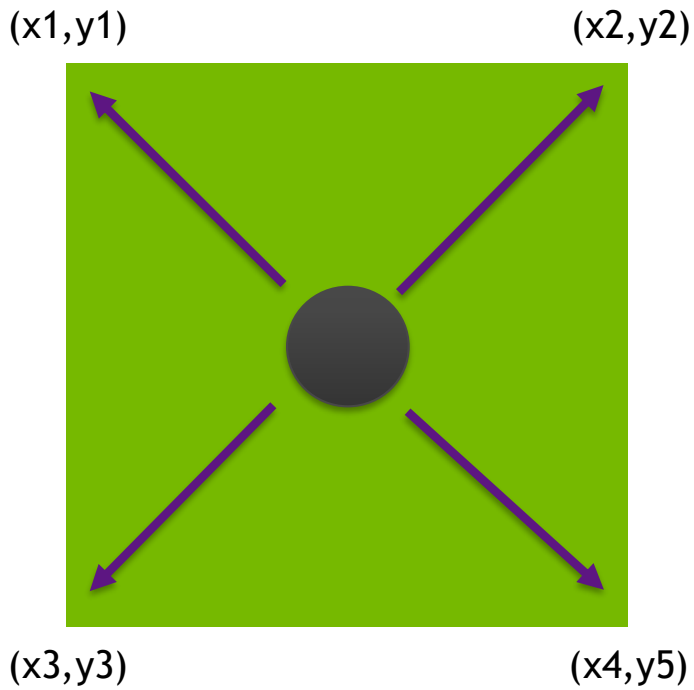
*Ref: https://www.particleincell.com/2011/vxb-rotation/

SCATTER PARTICLE INSTEAD OF GATHER

- We separate velocity direction and magnitude. Magnitude in FP32 while directions in FP16
- We pack velocity, magnetic field vectors into Tensor Core format. This is basically the scatter operation.
- The GEMM updates velocities and add them back to particle final velocity at a given time step in FP32



To use Tensor core, scatter properties of the particles and use WMMA to compute and assemble



Gather by interpolation forces from the cell vertices.

Scatter particle properties to nodes and add compute at nodes.

BORIS METHOD

*Boris method is the *de facto* standard for particle pushing in plasma simulation codes
It is an explicit technique

The following equations summarize Boris method

In the absence of Electric Field. V+ acts as velocity update. Electric field can be easily added.

$$\frac{v^+ - v^-}{\Delta t} = \frac{q}{2m}(v^+ + v^-) \times B$$

$$v' = v^- + v^- \times t \qquad t = (q\,B/m)\Delta t/2$$

$$v^+ = v^- + v' \times s \qquad s = \frac{2t}{1+t^2}$$

$$v' = v^- + -1 * (\,\mathrm{f1(B)} \times\ v^-)$$

$$v^+ = v^- + -1\ * (\,\mathrm{f2(B)} \times\ v')$$

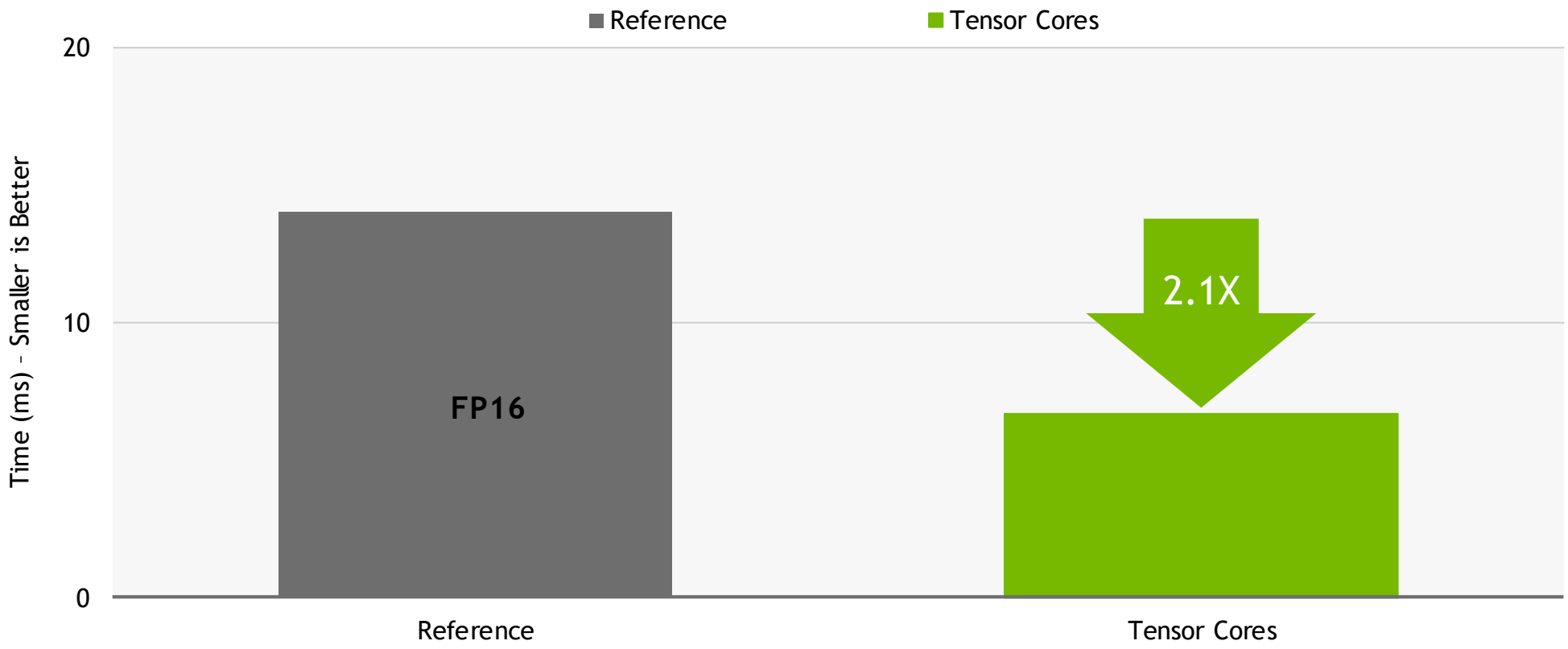
B1	B2	B3	B4
B5	B6	B7	B8

×

P1 v1	…	P32 v1
P1 v2	…	P32 v2
P1 v3	…	P32 v3
P1 v4	…	P32 v4

*Ref: https://www.particleincell.com/2011/vxb-rotation/

MINI-APP PICTC PERFORMANCE COMPARISON

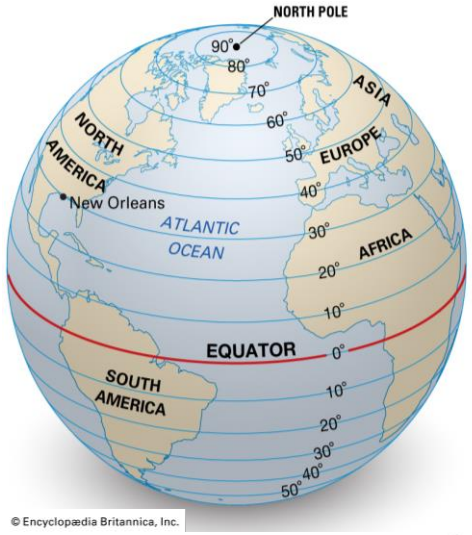


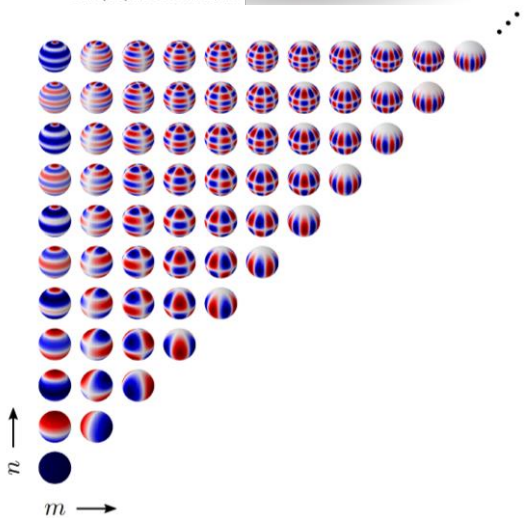
Source: CUDA 10.1, Summit

https://github.com/vishalmehta1991/pictc

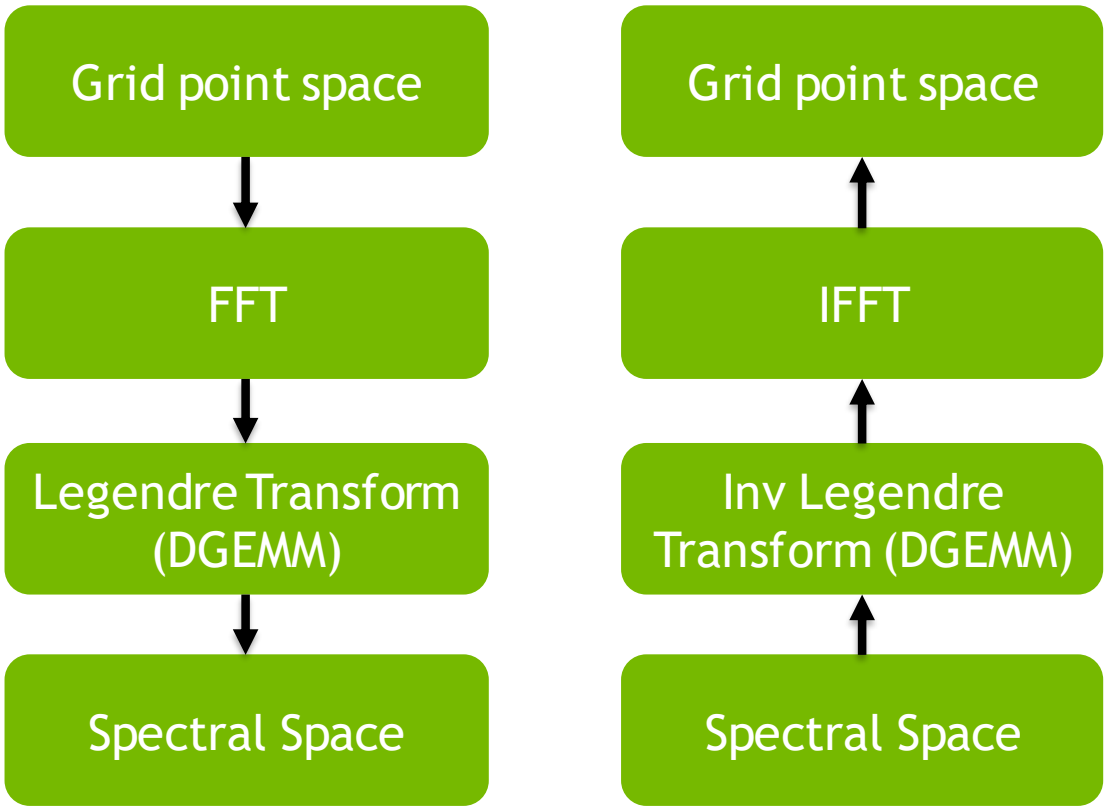
SPHERICAL HARMONICS IN IFS (WEATHER & CLIMATE)

SPHERICAL HARMONICS IN IFS



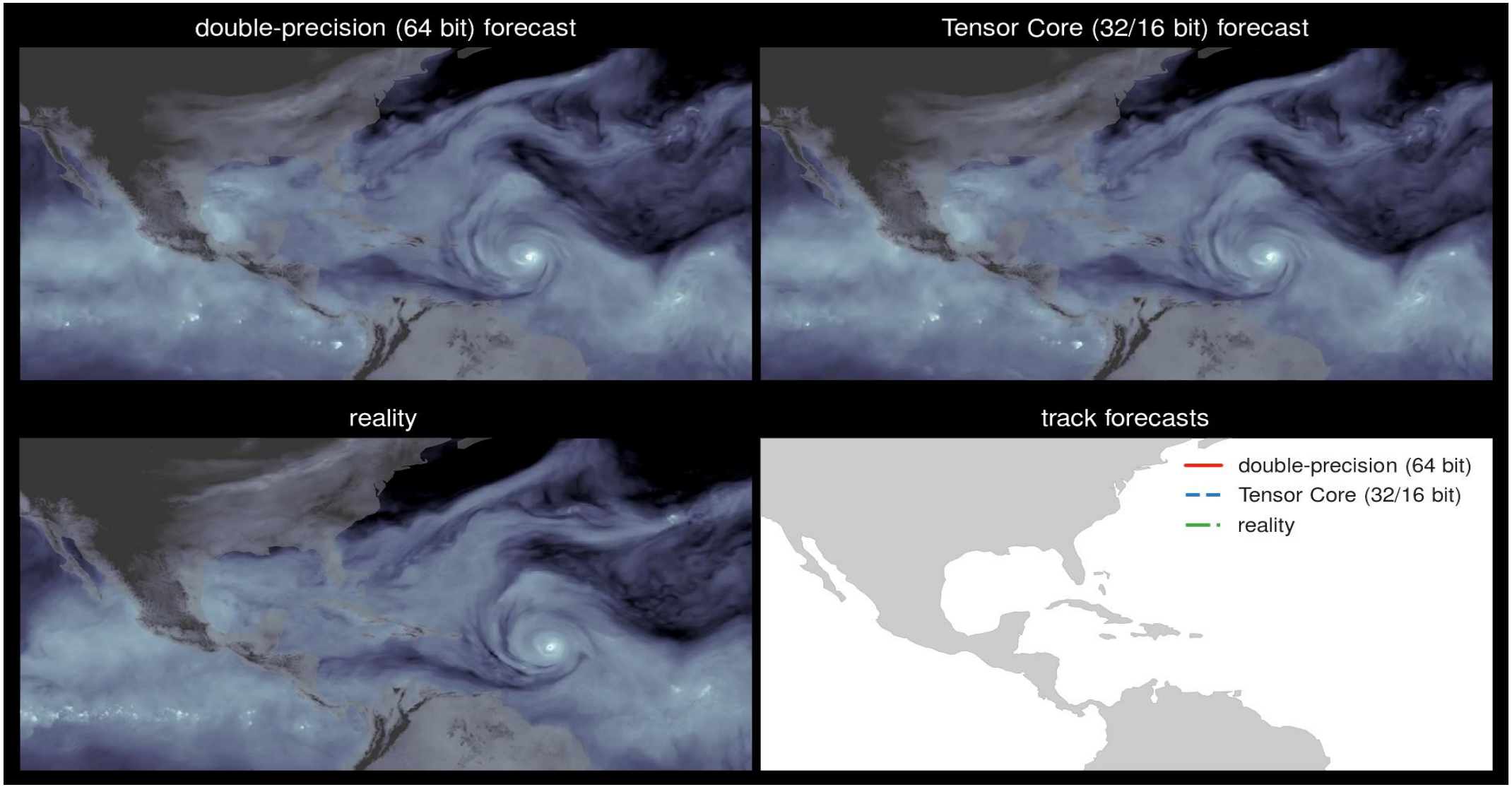


Spherical harmonics are eigen functions of the Laplacian in spherical co-ordinates.



https://doi.org/10.1145/3324989.3325711

FORECAST FOR HURRICANE IRMA



SUMMARY

- When used appropriately Tensor Cores can achieve as much as an 8X performance increase.
- A variety of High and Low-level entry points are available for programming Tensor Cores
- **Rethinking data layouts, mixed precision and algorithmic patterns is the key to Tensor Core utilization.**

ADDITIONAL RESOURCES

CUTLASS Basics - http://on-demand.gputechconf.com/gtc/2018/presentation/s8854-cutlass-software-primitives-for-dense-linear-algebra-at-all-levels-and-scales-within-cuda.pdf

cuTensor & CUTLASS - https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/S9593/

cuBLAS - https://docs.nvidia.com/cuda/cublas/index.html

Mixed Precision Training Guide - https://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html

CUDA C Programming Guide (WMMA) - https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#wmma

PICTC Mini-APP - https://github.com/vishalmehta1991/pictc

PTX ISA - https://docs.nvidia.com/cuda/parallel-thread-execution/index.html

CUDA Tensor Core Sample - https://docs.nvidia.com/cuda/cuda-samples/index.html#cuda-tensor-core-gemm