# Introduction to cuBLAS

Goran Frehse
ENSTA Paris

Disclaimer:

The following slides are excerpts with slight adaptations from the excellent course by Al Barr, Aadyot Bhatnagar, Tyler Port, Bobby Abrahamson

http://courses.cms.caltech.edu/cs179/

# BLAS and cuBLAS

- https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms
- BLAS defines a set of common functions for to scalars, vectors, and matrices.
- Libraries that implement it exist in almost all major languages.
- different functions for different number types

- **cuBLAS = BLAS function library for CUDA**

Taken from http://courses.cms.caltech.edu/cs179/

# BLAS Levels

- Level 1: Scalar and Vector, Vector and Vector operations, $\gamma \rightarrow \alpha\chi + \gamma$
- Level 2: Vector and Matrix operations, $\gamma \rightarrow \alpha\mathbf{A}\chi + \beta\gamma$
- Level 3: Matrix and Matrix operations, $\mathbf{C} \rightarrow \alpha\mathbf{AB} + \beta\mathbf{C}$

    - Scalars: $\alpha$, $\beta$
    - Vectors: $\chi$, $\gamma$
    - Matrices: **A**, **B**, **C**

Taken from http://courses.cms.caltech.edu/cs179/

# cuBLAS Level 1

https://docs.nvidia.com/cuda/cublas

# cuBLAS Level 2

https://docs.nvidia.com/cuda/cublas

# cuBLAS Level 3

https://docs.nvidia.com/cuda/cublas

# The various cuBLAS types

- Every function exists in different versions for different number types
  - S, s : single precision (32 bit) real float
  - D, d : double precision (64 bit) real float
  - C, c : single precision (32 bit) complex float (implemented as a float2*)
  - Z, z : double precision (64 bit) complex float
  - H, h : half precision (16 bit) real float

* float2 is a struct of two floats

Taken from http://courses.cms.caltech.edu/cs179/

# cuBLAS function types

- cublasIsamax -> cublas I s amax
  - I : stands for index.
  - s : this is the single precision float variant of the isamax operation
  - amax : finds a maximum
- cublasSgemm → cublas S gemm
  - S : single precision real float
  - gemm : general matrix-matrix multiplication
- cublasHgemm
  - H : half precision real float
  - gemm : general matrix-matrix multiplication
- cublasDgemv → cublas D gemv
  - D : double precision real float
  - gemv : general matrix vector multiplication

Taken from http://courses.cms.caltech.edu/cs179/

# Using cuBLAS

- https://developer.nvidia.com/sites/default/files/akamai/cuda/files/Misc/mygpu.pdf
  - cuBLAS examples in different implementations
- http://docs.nvidia.com/cuda/cublas/index.html the official NVIDIA docs.
- Include the header “cublas_v2.h” and link the library with “-lcublas”
- must create a handle before using cuBLAS functions:

```
cublasHandle_t handle;
stat = cublasCreate(&handle);
    if (stat != CUBLAS_STATUS_SUCCESS) {
        printf ("CUBLAS initialization failed\n");
        return EXIT_FAILURE;
    }
… use cuBLAS functions …
cublasDestroy(handle);
```

# Example: Matrix-Matrix Multiplication

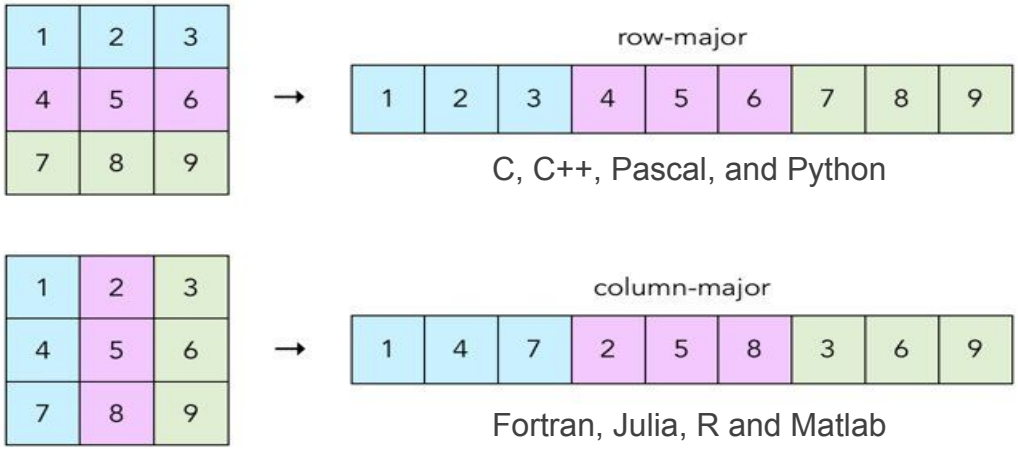
```
cublasSgemm(h,transpA,transpB,m,n,k,&alpha,&A,lda,&B,ldb,&beta,&C,ldc)
```

implements C = α op(A) op(B) + β C

where op(A) is A if transpA = CUBLAS_OP_N
transpose(A) if transpA = CUBLAS_OP_T
...similar for op(B) …

lda,ldb,ldc = number of rows of A,B,C
m,n,k are according to the dimensions of op(A) : m × k , op(B): k × n, C: m × n

# CuBLAS - Column Major




For efficiency, traverse matrices the way they are stored.

https://craftofcoding.wordpress.com/2017/02/03/column-major-vs-row-major-arrays-does-it-matter/

# Array Indexing

- use an indexing macro:

```
#define IDX2C(i,j,ld) (((j)*(ld))+(i))
```

Where “i” is the row, “j” is the column, and “ld” is the leading dimension.

In column major storage “ld” is the **number of rows**.

Taken from http://courses.cms.caltech.edu/cs179/