

GPUs for HPC

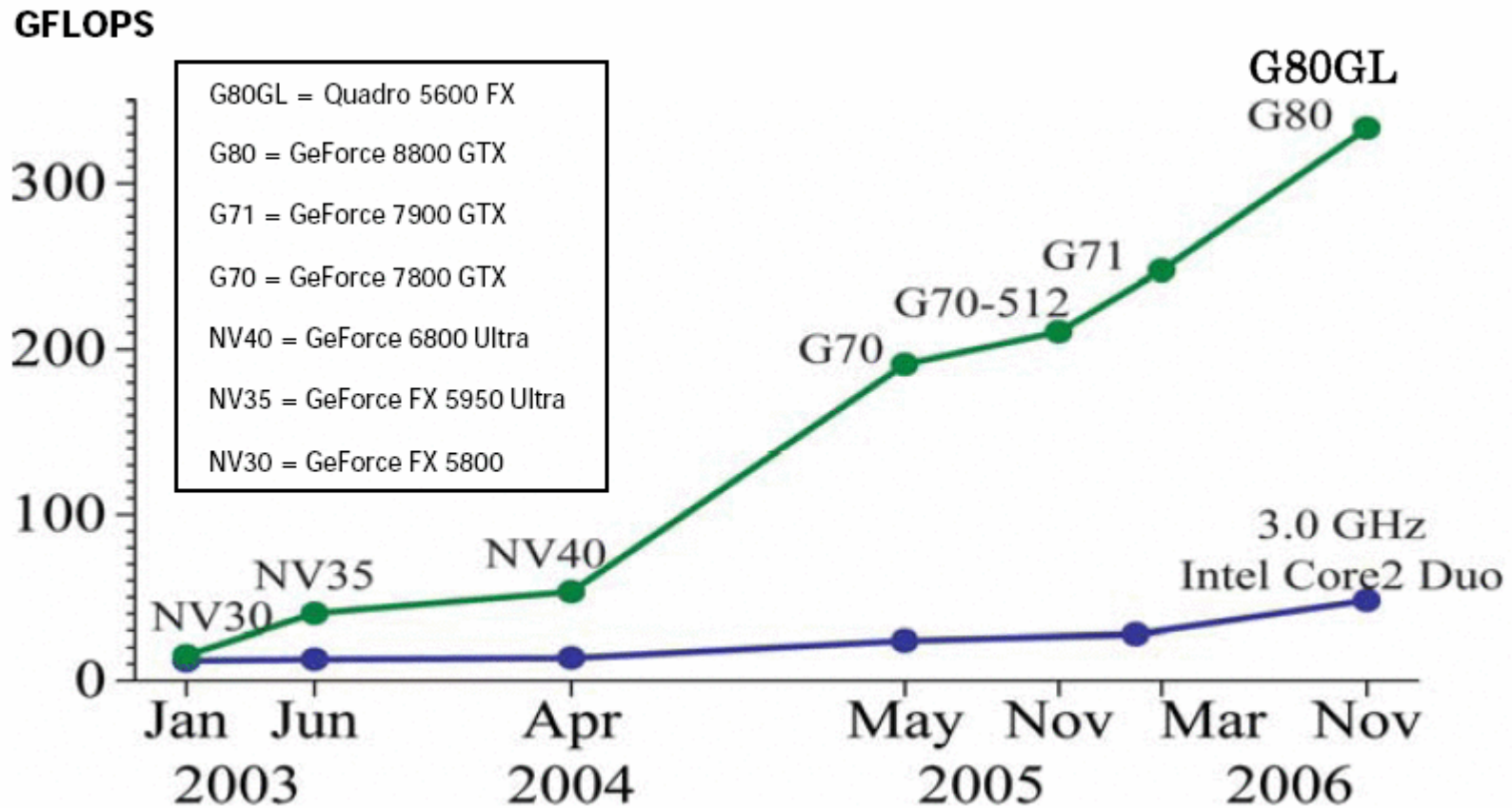
NVIDIA's Compute Unified Device Architecture (CUDA)

Stan Tomov

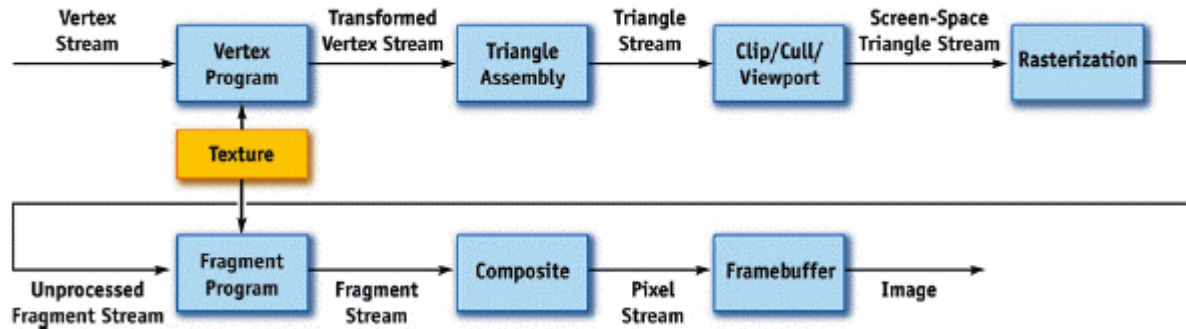
03/05/2008

specifications and graphs taken from
CUDA Programming Guide Version 1.0
(references on: http://www.nvidia.com/object/cuda_get.html)

CPUs and GPUs



Computing on the GPU



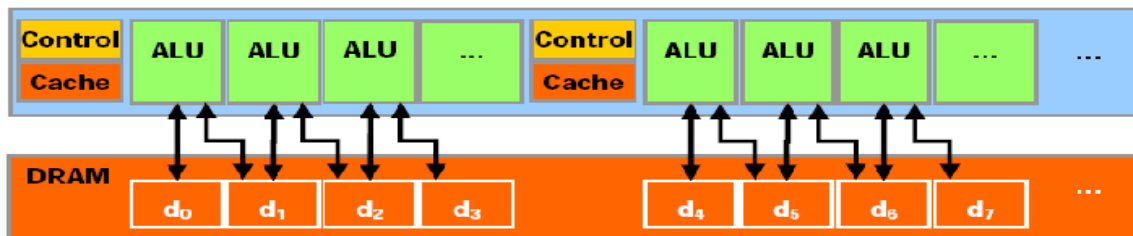
From **non-programmable hardware**
(fixed-function pipeline)

To **simple streaming**
(data viewed as streams, computation as kernels)

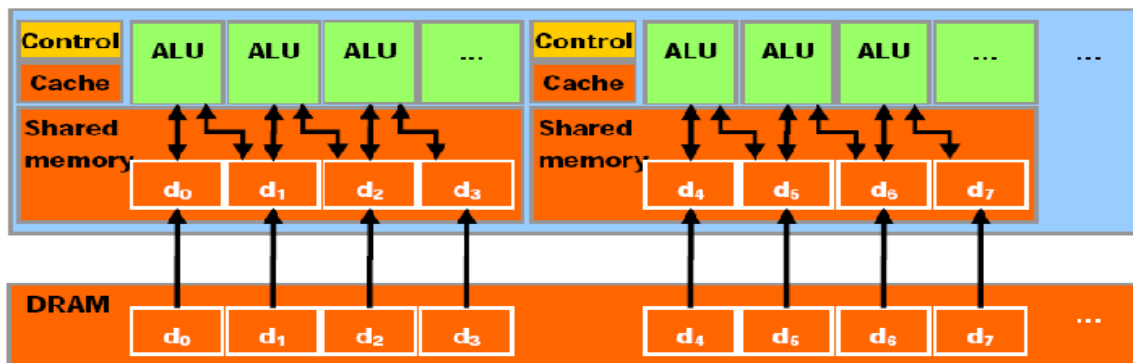
more powerful streaming
(gather, scatter, reduction, etc
more CPU features)

CUDA (NVIDIA) / CTM (ATI)
* general DRAM addressing
(powerful gather, scatter, etc.)
* very fast shared memory
* to have DP arithmetic by end of year

... ?
* fusion with CPU?

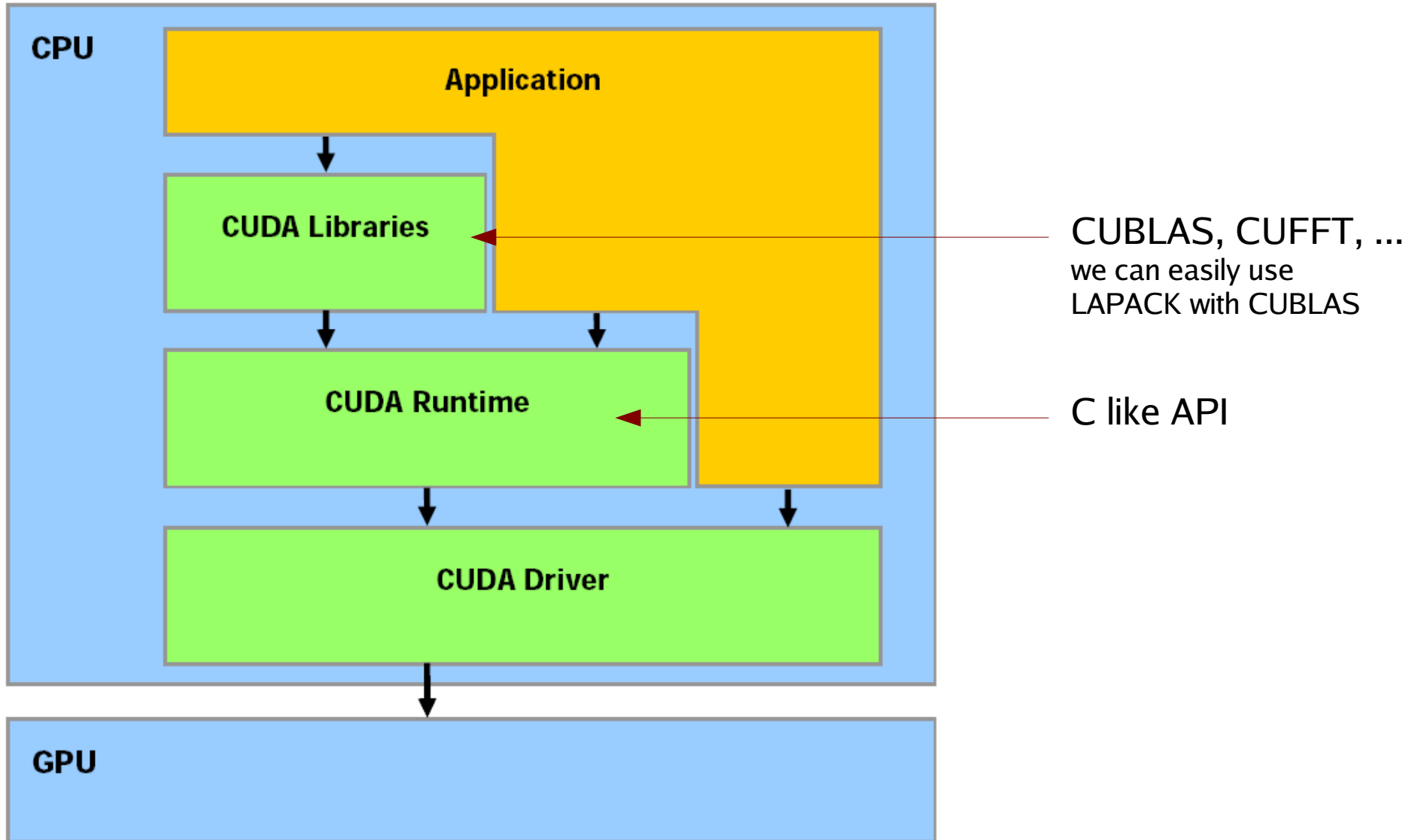


Without shared memory



With shared memory

CUDA



Programming model

A highly multithreaded coprocessor

* thread block

(a batch of threads with fast shared memory executes a kernel)

* Grid of thread blocks

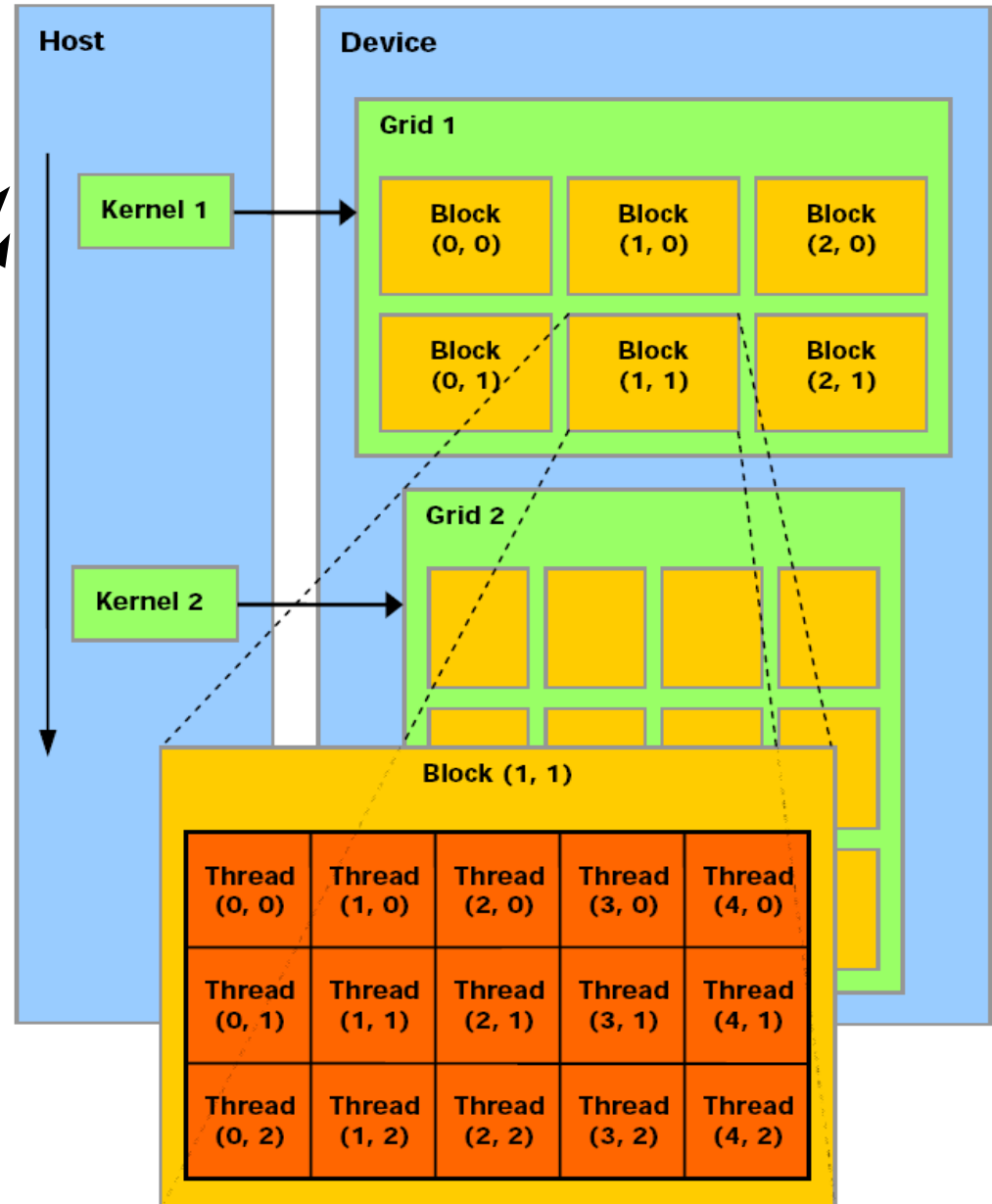
(blocks of the same dimension, grouped together to execute the same kernel;
reduces thread cooperation)

```
// set the grid and thread configuration
Dim3 dimBlock(3,5);
Dim3 dimGrid(2,3);

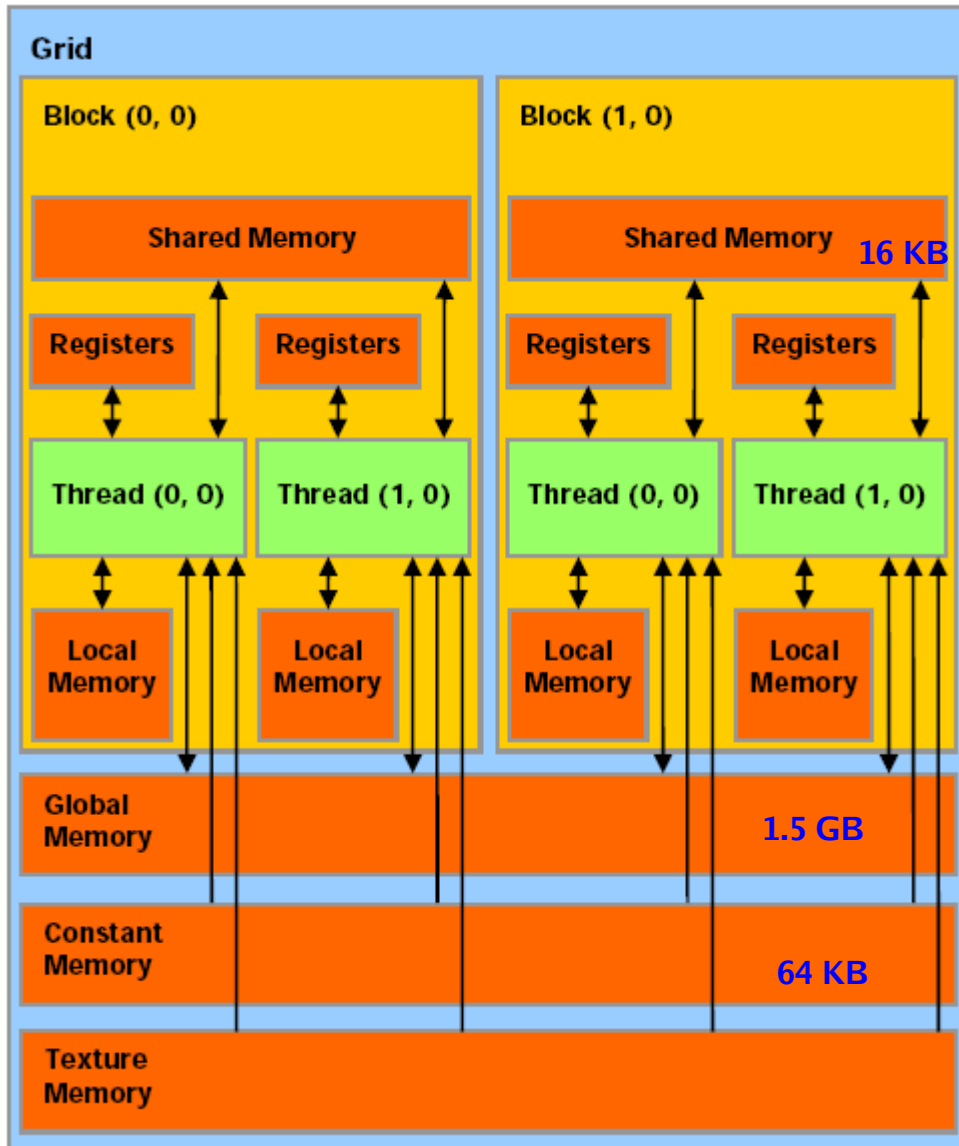
// Launch the device computation
MatVec<<<dimGrid, dimBlock>>>( ... );
```

```
__global__ void MatVec( ... ) {
// Block index
int bx = blockIdx.x;
int by = blockIdx.y;

// Thread index
int tx = threadIdx.x;
int ty = threadIdx.y;
...
}
```



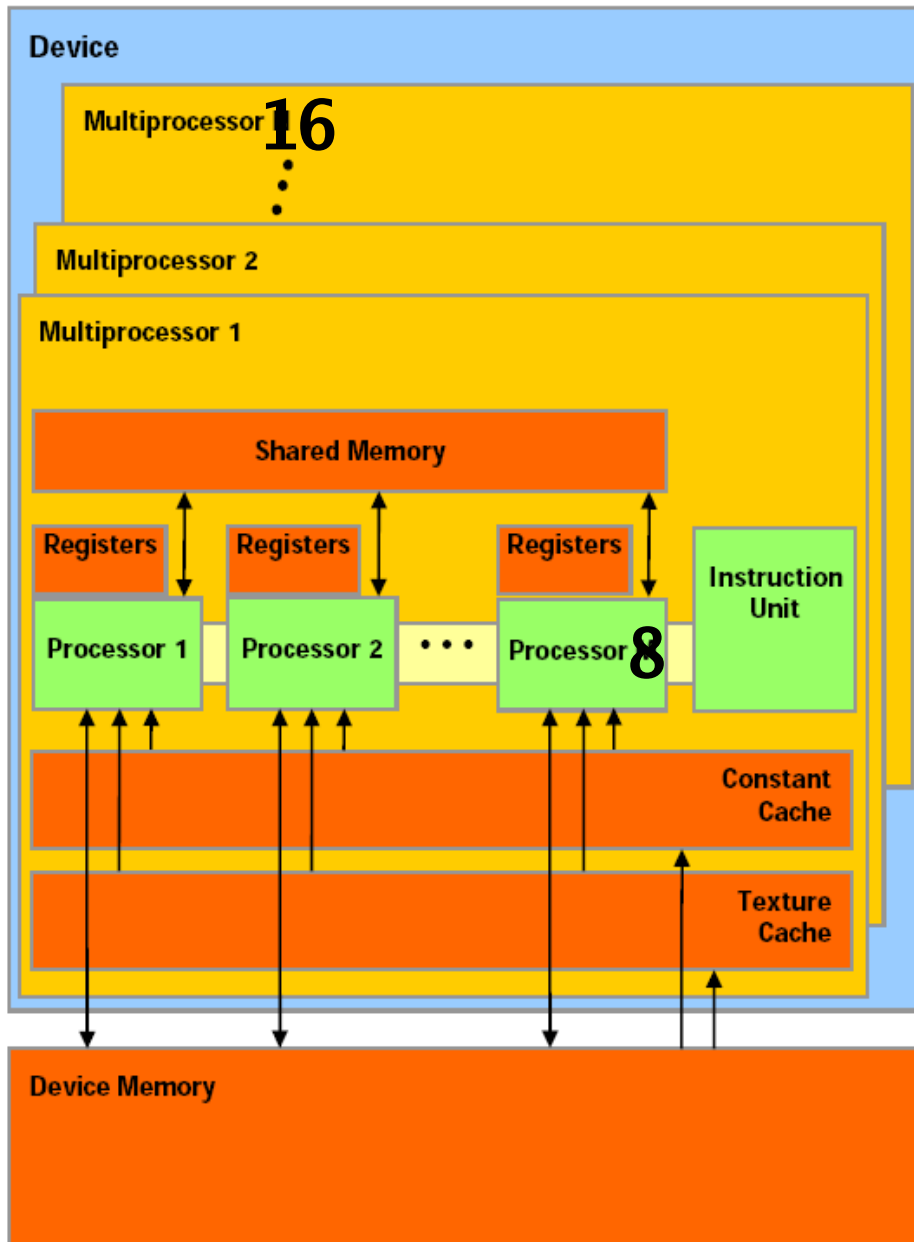
Memory model



We have
Quadro FX 5600

Theoretical memory bandwidth: **76.8 GB/s**
Interface : PCI Express x 16
* up to **4GB/s** peak per direction
* up to **8BG/s** concurrent bandwidth

Hardware Model



Quadro FX 5600

Some numbers:

- processors: 128 (total)
- registers : 8192 per block
- warp size : 32
- max threads per block: 512

Execution of Grid of thread blocks:

- * one or more blocks are executed on multiprocessor using time slicing
- * each block is split into group of threads (called *warps*)
- * a block is processed by only 1 multiprocessor
- * issue order of warps within a block is undefined (but execution can be synchronized)
- * issue order of blocks within a grid is undefined
 - thread of different blocks (on the same grid) can not safely communicate through global memory
- + many other issues ... (in the manual)

Performance issues

- **Instruction throughput**

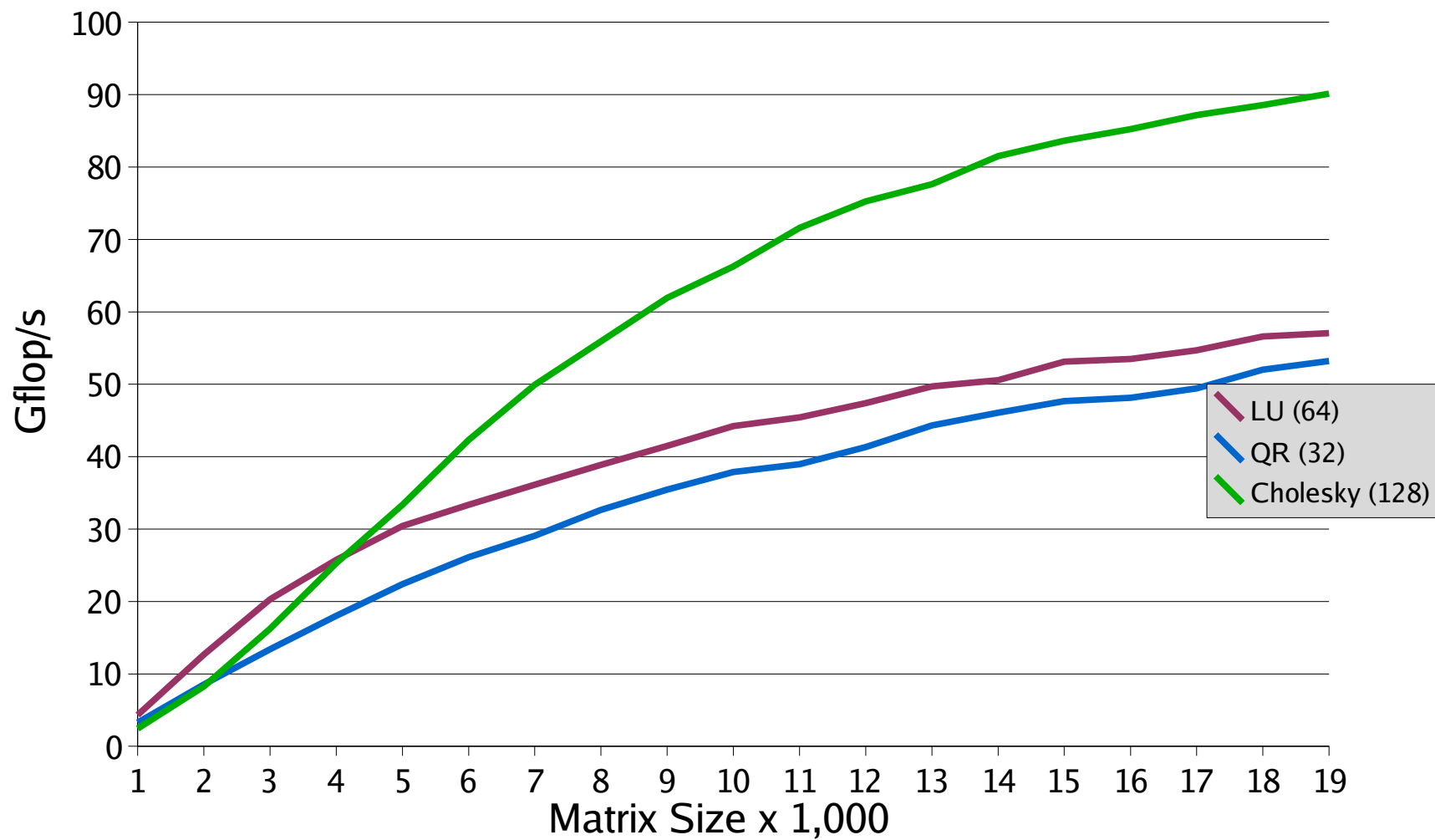
- Arithmetic instructions (some more 'expensive' than others)
- Control flow (if, switch, do, for, while can significantly impact on performance)
- Memory instructions (400..600 cycles in memory latency to read/write to global memory; 4 cycles for shared can be hidden by thread scheduler if there is enough computation to be overlapped with)
- Thread synchronization (4 cycles to issue for a warp if no thread has to wait)

- **Memory bandwidth**

- Load data to shared memory -> synchronize with other block threads to have safe/fast read/writes -> process -> synchronize again for updates -> write result back
- Global memory (aligning data for faster access; not cached)
- Constant memory (cached)
- Texture memory (cached; optimized for 2D spacial locality)
- Shared memory (divided into 16 banks; can be accessed simultaneously; successive 32-bit words go into successive banks; bandwidth: 32 bits/2 cycles)

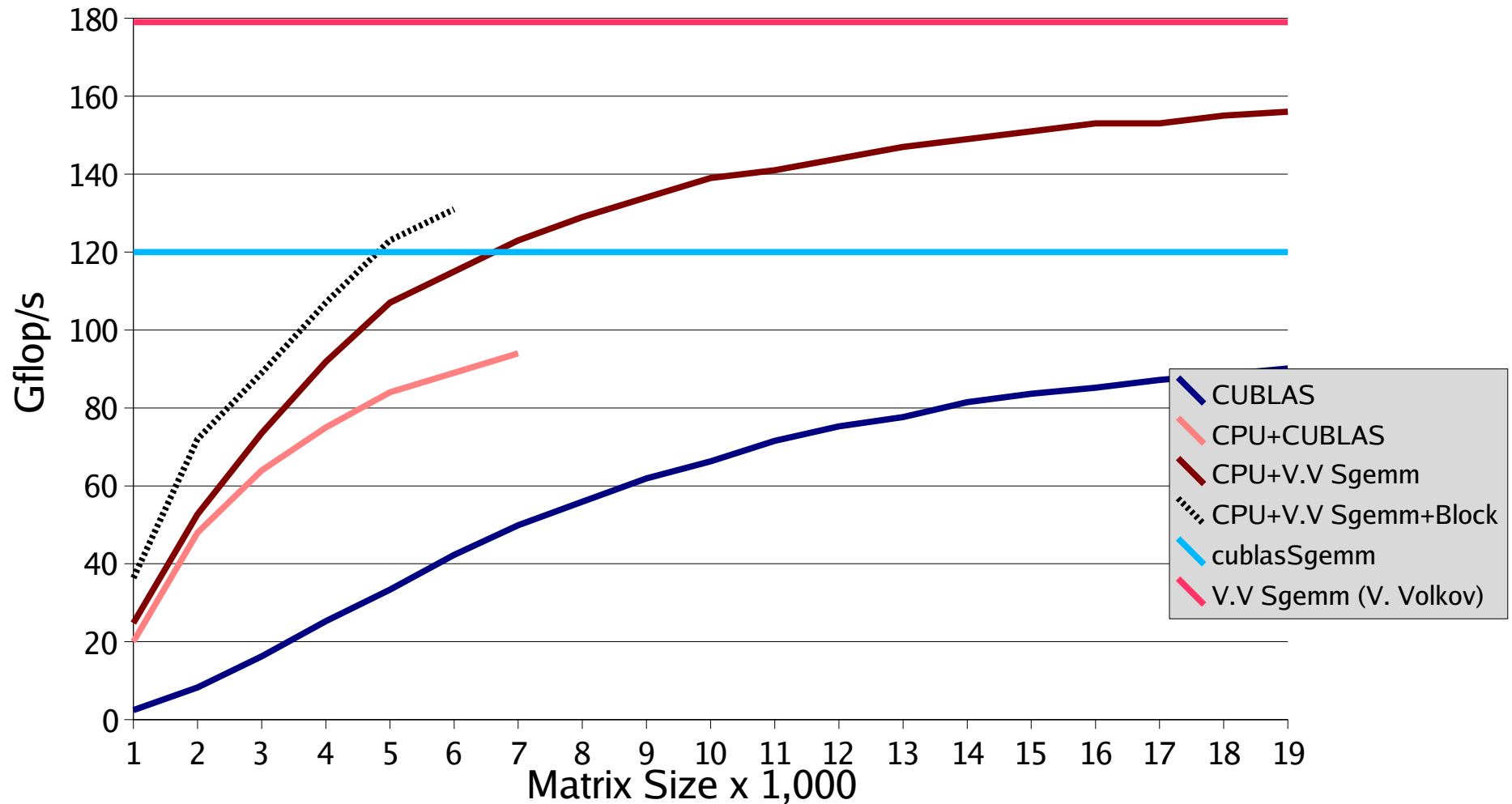
- **Other** (tune number of threads/block; data transfer between CPU and GPU, etc ...)

Performance of NVIDIA Quadro FX 5600 on LAPACK + CUDA BLAS



Performance of NVIDIA Quadro FX 5600 on LAPACK Cholesky + CUDA BLAS

(single precision; theoretical GPU peak 346 Gflop/s)



Performance of sgemm versions (on NVIDIA's Quadro FX 5600)

