

# MISC. CUDA TOPICS

2D arrays, performance profiling

# 2D ARRAYS IN CUDA

```
// host code
int A[10][20] = ...;
A[5][6] = 17;
cudaMemcpy(d_A, A, ...);

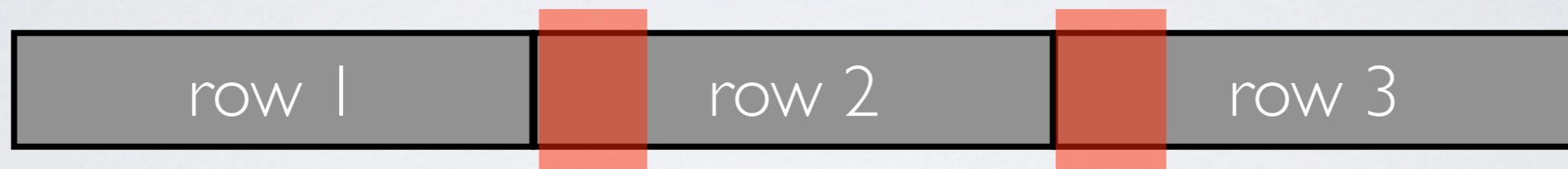
// device code
__device__ kernel(d_A) {
    d_A[5][6] = 17;
}
```

# 2D ARRAYS IN CUDA

- 2 problems
  - don't know array bounds: `d_A` is an `int*`
  - rows beyond the first may not be optimally aligned

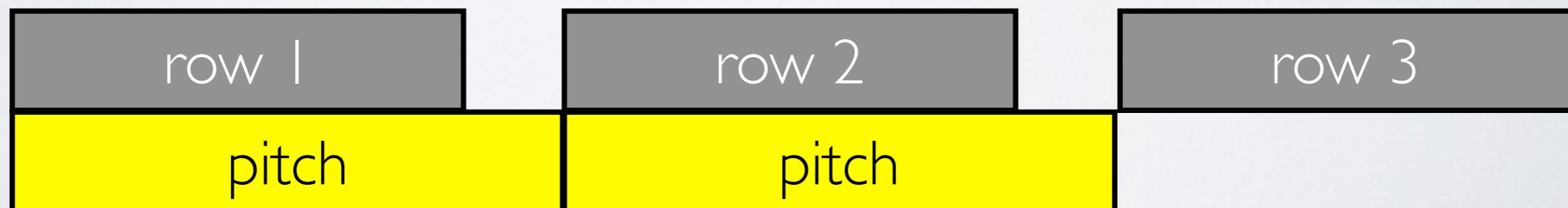
# 2D ARRAYS IN CUDA

Conventional C memory layout



**misalignment can harm global memory coalescing**

CUDA **pitched** memory



# CUDA PITCHED MEMORY

- 2D array indexing involves row, column and **pitch**

```
cudaError_t cudaMalloc3D(cudaPitchedPtr* pitchedDevPtr,  
                        cudaExtent extent)
```

```
cudaExtent make_cudaExtent(  
    size_t w, // bytes  
    size_t h, size_t d) // elements
```

- How do we index a pitched 2D array?

```
int* i = (int*)((char*)BaseAddr + Row * Pitch) + Col;
```

# CUDA PITCHED MEMORY

- Must use **pitch-aware** memcpy/memset

```
cudaError_t cudaMemcpy2D(  
    void* dst,  
    size_t dpitch, // bytes  
    const void* src,  
    size_t spitch, size_t width, // bytes  
    size_t height, // rows  
    cudaMemcpyKind kind)
```

# CUDA PITCHED MEMORY GOTCHAS

- **pitch is always specified in bytes**
- height/depth are specified **in elements**
  - in terms of rows/2D slices, respectively
- `cudaMallocArray` and friends use the Texture Cache
  - optimized layout for graphics textures that uses a space-filling curve for memory layout
  - [https://en.wikipedia.org/wiki/Z-order\\_curve](https://en.wikipedia.org/wiki/Z-order_curve)

# WHEN CAN I STOP OPTIMIZING?

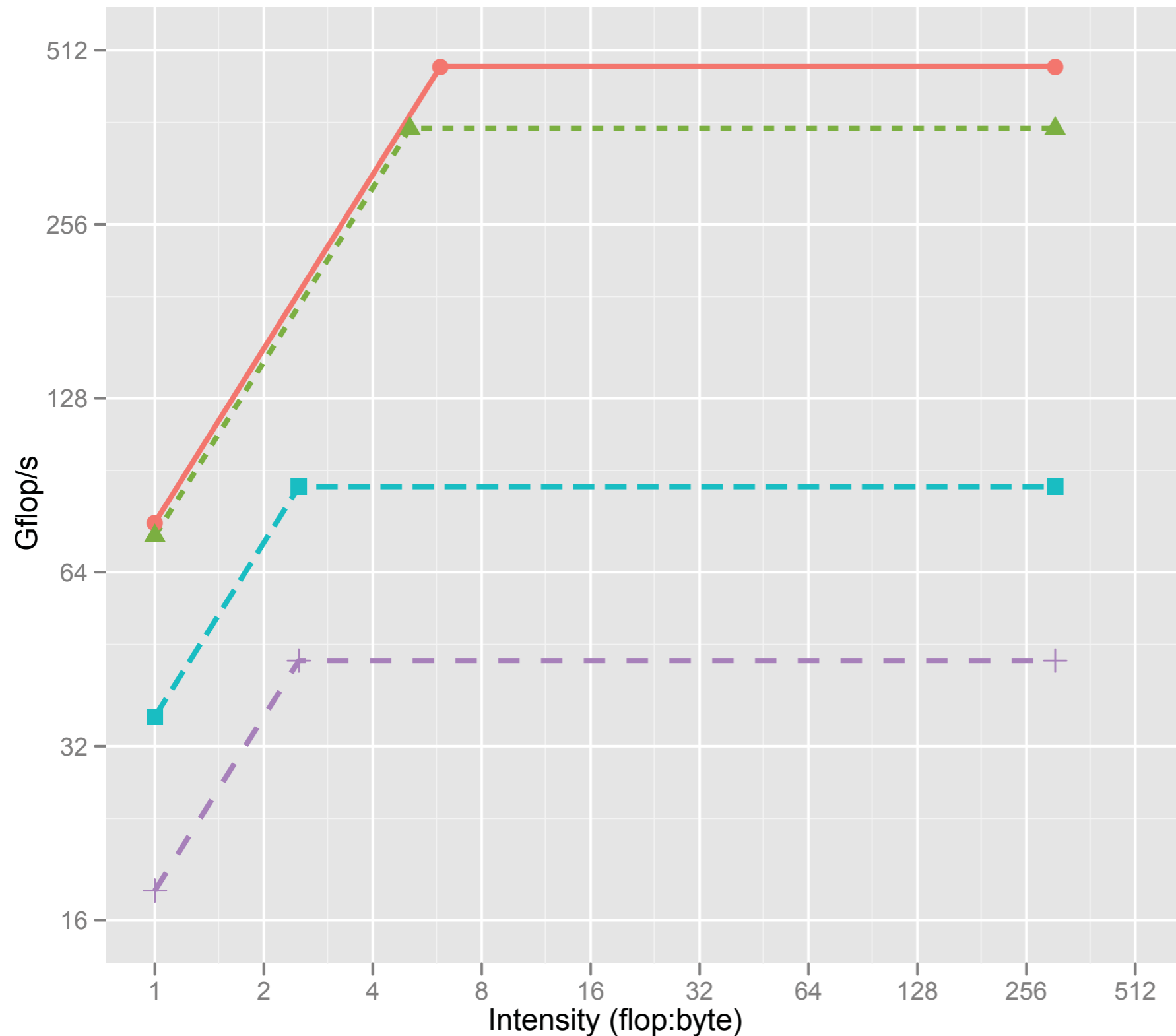
- Our GPUs: Nvidia GK104 (~GeForce 600)
- (global) memory bandwidth: **160 GB/s**
- compute bandwidth: 1536 “CUDA cores” x 800MHz = **1.2 TFlops** (~**2.4TFlops** with FMA)
- are we *memory* or *compute* limited?



# ARITHMETIC INTENSITY

- GK104 ideal flop-to-byte ratio =  $1200/160 = \mathbf{7.5}$
- what is blurGlobal's behavior?
  - 5600 flops per thread
  - 450 mops per thread (4B each!)
  - $\sim 3.1$  flop-to-byte ratio

# ROOFLINE ANALYSIS



Kim et al., *Performance Analysis and Tuning for General Purpose Graphics Processing Units*

**Platform**

- Fermi
- C1060
- Nehalem x 2
- Nehalem

Williams et al., *Roofline: An insightful visual performance model for multicore architectures. Communications of the ACM, 52(4):65, April 2009*

# CUDA PERFORMANCE PROFILING

- <demo in AWS>

# HOW FAST IS blurShared?

- $4096 \times 3072 \text{ pixels} = 12.6\text{M pixels} * 5600 \text{ flops/pixel} = 70 \text{ Gflop}$
- blurShared runs in 50ms = 0.05s
- $70 \text{ Gflop} / 0.05\text{s} = \mathbf{1.4 \text{ Tflops}}$
- not too shabby!

# WHEN CAN I STOP OPTIMIZING?

- max Flops/Fliops depends on what instructions you/compiler use
- memory bandwidth depends on which memory you use