

# Visual Studio CPU Debug and Nsight CUDA Debug

ECE 277

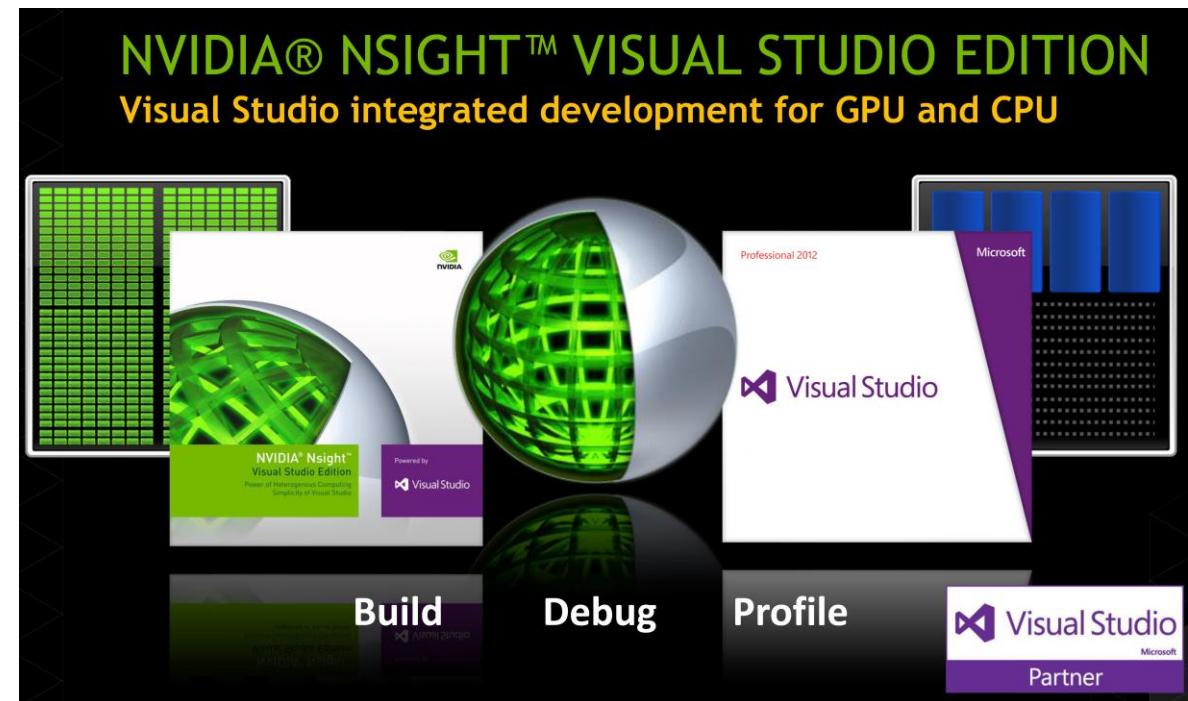
Cheolhong An

Integrated into VS.

# Visual Studio + Nsight VS edition

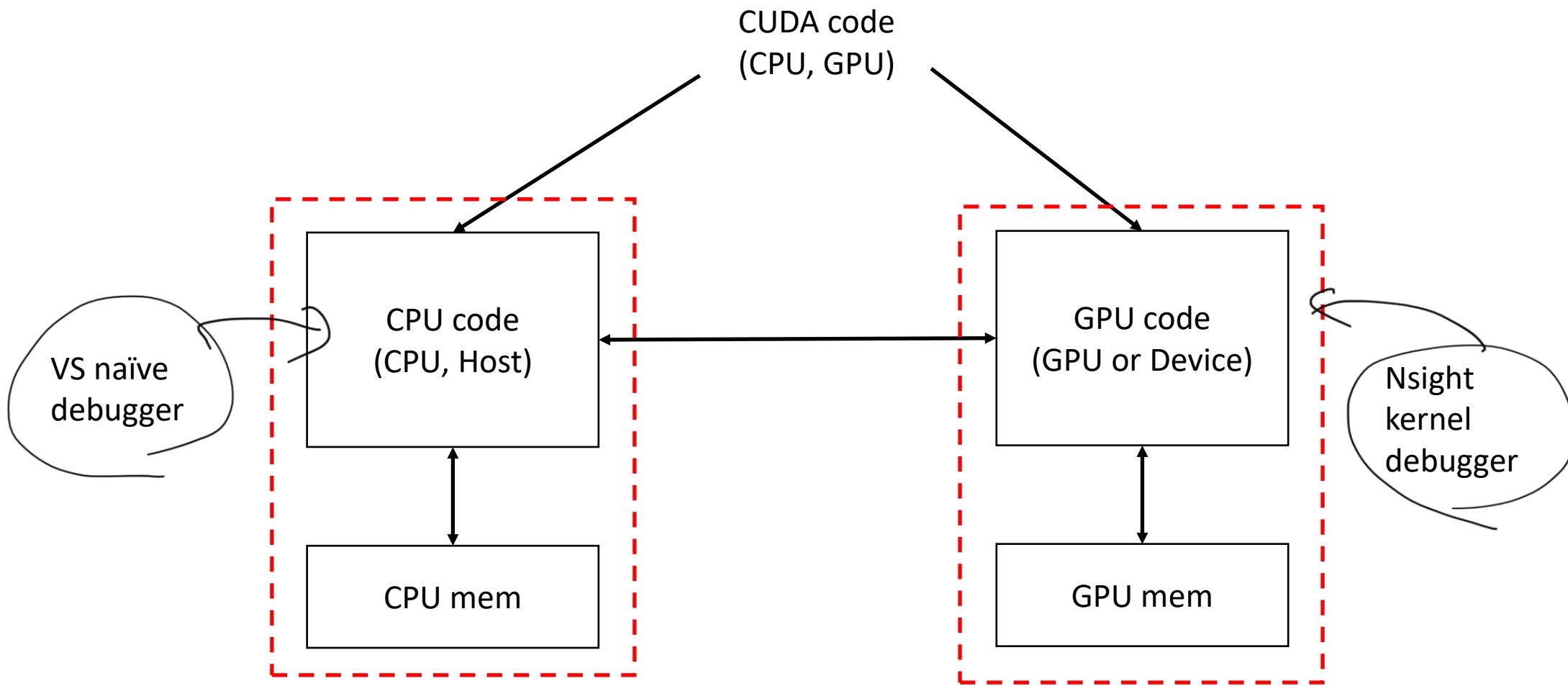
- CUDA debug (**Nsight VS**) is integrated into Visual Studio for debug and profile
- However, **separate launches** are required for CUDA kernel debug (**Nsight**) and CPU debug (**VS naïve**)

⇒ debug separately.



# CUDA programming

- support both cpu and gpu (Heterogeneous)



# VS Debug: CPU (1/2)

- 1) Compile a project with “Debug” mode
- 2) Add a breakpoint  
click the gray area and a red dot appears
- 3) Run a program

0=71  
21=0  
21=0  
21=0  
(break point)

1)

3)

2)

Visual Studio Screenshot:

- File Edit View Project Build Debug Team Nsight Tools Test Analyze Window Help
- sumMatrixOnGPU-2D-grid-2D-block.cu sumMatrixOnGPU-1D-grid-1D-block.cu bandwidthTest.cu
- Local Windows Debugger
- Solution Explorer
- Properties
- C++
- Python 3.6 Interactive Find Results 1 Error List Output Find Symbol Results
- Ready Ln 147 Col 28 Ch 28 INS
- 4

```
136 dim3 block(dimx, dimy);
137 dim3 grid((nx + block.x - 1) / block.x, (ny + block.y - 1) / block.y);
138
139 iStart = seconds();
140 sumMatrixOnGPU2D<<<grid, block>>>(d_MatA, d_MatB, d_MatC, nx, ny);
141 CHECK(cudaDeviceSynchronize());
142 iflaps = seconds() - iStart;
143 printf("sumMatrixOnGPU2D <<<(%d,%d), (%d,%d)>>> elapsed %f sec\n", grid.x,
144     grid.y,
145     block.x, block.y, iflaps);
146 // check kernel error
147 CHECK(cudaGetLastError());
148
149 // copy kernel result back to host side
150 CHECK(cudaMemcpy(gpuRef, d_MatC, nBytes, cudaMemcpyDeviceToHost));
151
152 // check device results
153 checkResult(hostRef, gpuRef, nx);
154
155 // free device global memory
156 CHECK(cudaFree(d_MatA));
157 CHECK(cudaFree(d_MatB));
158 CHECK(cudaFree(d_MatC));
159
160 // free host memory
161 free(h_A);
162 free(h_B);
163 free(hostRef);
164 free(gpuRef);
165
166 // reset device
167 CHECK(cudaDeviceReset());
```

Output:

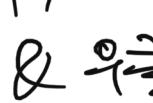
```
Show output from: Debug
c1_block2d_grid2d.exe (Win32) loaded C:\Program Files\NVIDIA Corporation\Nsight Visual Studio Edition 2019\1\Monitor\CommonInjection64\Nvda.Cuda.Injection.dll'. Cannot find or open the PDB file.
'c1_block2d_grid2d.exe' (Win32): Loaded 'C:\Windows\System32\nvapi64.dll'. Cannot find or open the PDB file.
'c1_block2d_grid2d.exe' (Win32): Loaded 'C:\Windows\System32\dxgi.dll'. Symbols loaded.
'c1_block2d_grid2d.exe' (Win32): Loaded 'C:\Windows\System32\dwmapi.dll'. Symbols loaded.
'c1_block2d_grid2d.exe' (Win32): Loaded 'C:\Windows\System32\uxtheme.dll'. Symbols loaded.
'c1_block2d_grid2d.exe' (Win32): Loaded 'C:\Windows\System32\wintrust.dll'. Symbols loaded.
'c1_block2d_grid2d.exe' (Win32): Loaded 'C:\Windows\System32\crypt32.dll'. Symbols loaded.
'c1_block2d_grid2d.exe' (Win32): Loaded 'C:\Windows\System32\msasn1.dll'. Symbols loaded.
The program '[36268] c1_block2d_grid2d.exe' has exited with code 0 (0x0).
```

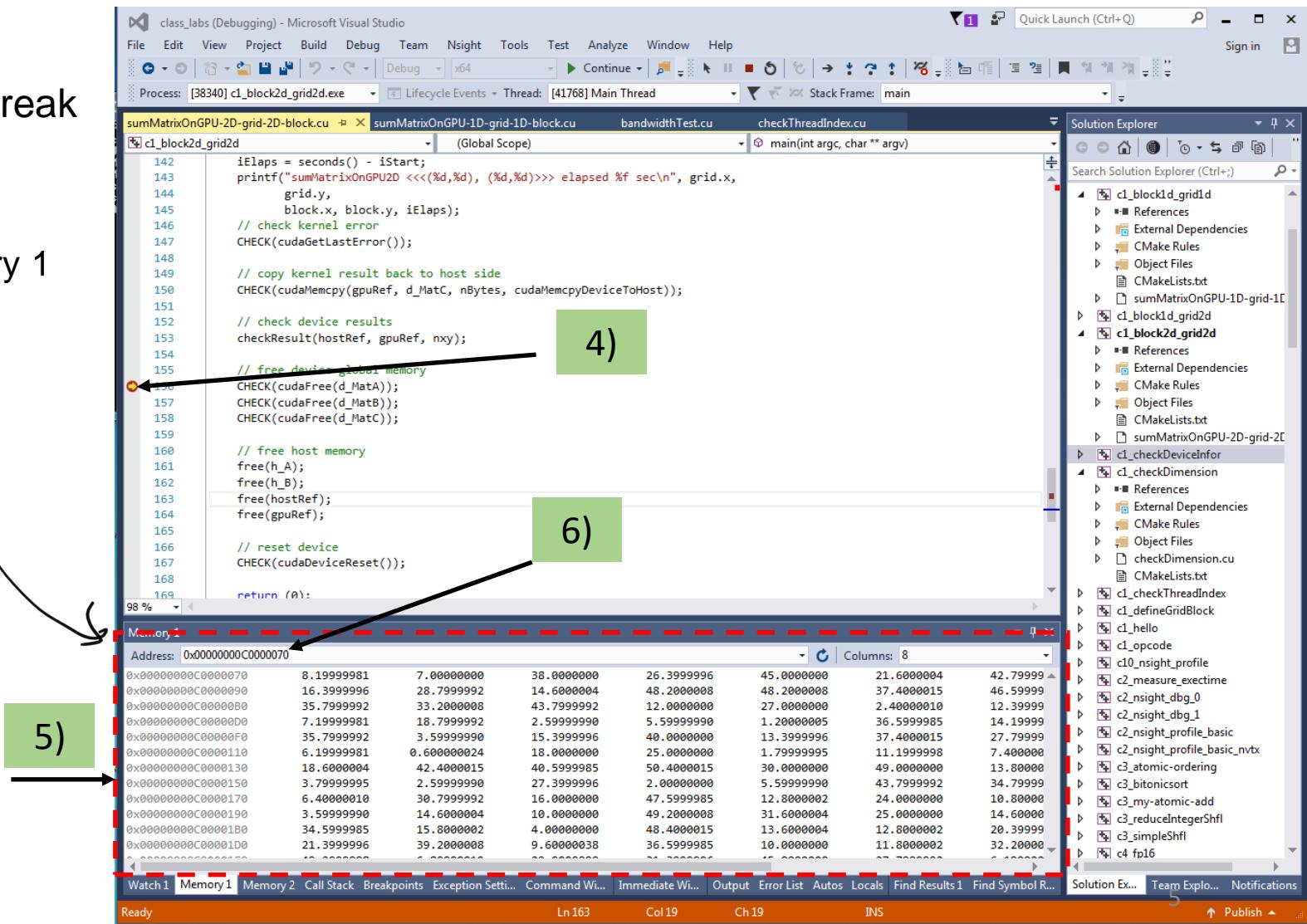
# VS Debug: CPU (2/2)

- 4) When the program stop at the break point, yellow arrow will appear
- 5) Open “Memory 1” window  
Debug->Windows->Memory->Memory 1
- 6) View memory contents

Type a point variable name into the address window

(copy or drag-drop)

&  (byte type offset { })



4)

5)

6)

Code in c1\_block2d\_grid2d.cu:

```
iElaps = seconds() - iStart;
printf("sumMatrixOnGPU2D <<(%d,%d), (%d,%d)>> elapsed %f sec\n", grid.x,
       grid.y,
       block.x, block.y, iElaps);
// check kernel error
CHECK(cudaGetLastError());

// copy kernel result back to host side
CHECK(cudaMemcpy(gpuRef, d_MatC, nBytes, cudaMemcpyDeviceToHost));

// check device results
checkResult(hostRef, gpuRef, nxny);

// free device global memory
CHECK(cudaFree(d_MatA));
CHECK(cudaFree(d_MatB));
CHECK(cudaFree(d_MatC));

// free host memory
free(h_A);
free(h_B);
free(hostRef);
free(gpuRef);

// reset device
CHECK(cudaDeviceReset());

return (0);
```

Memory 1 window content:

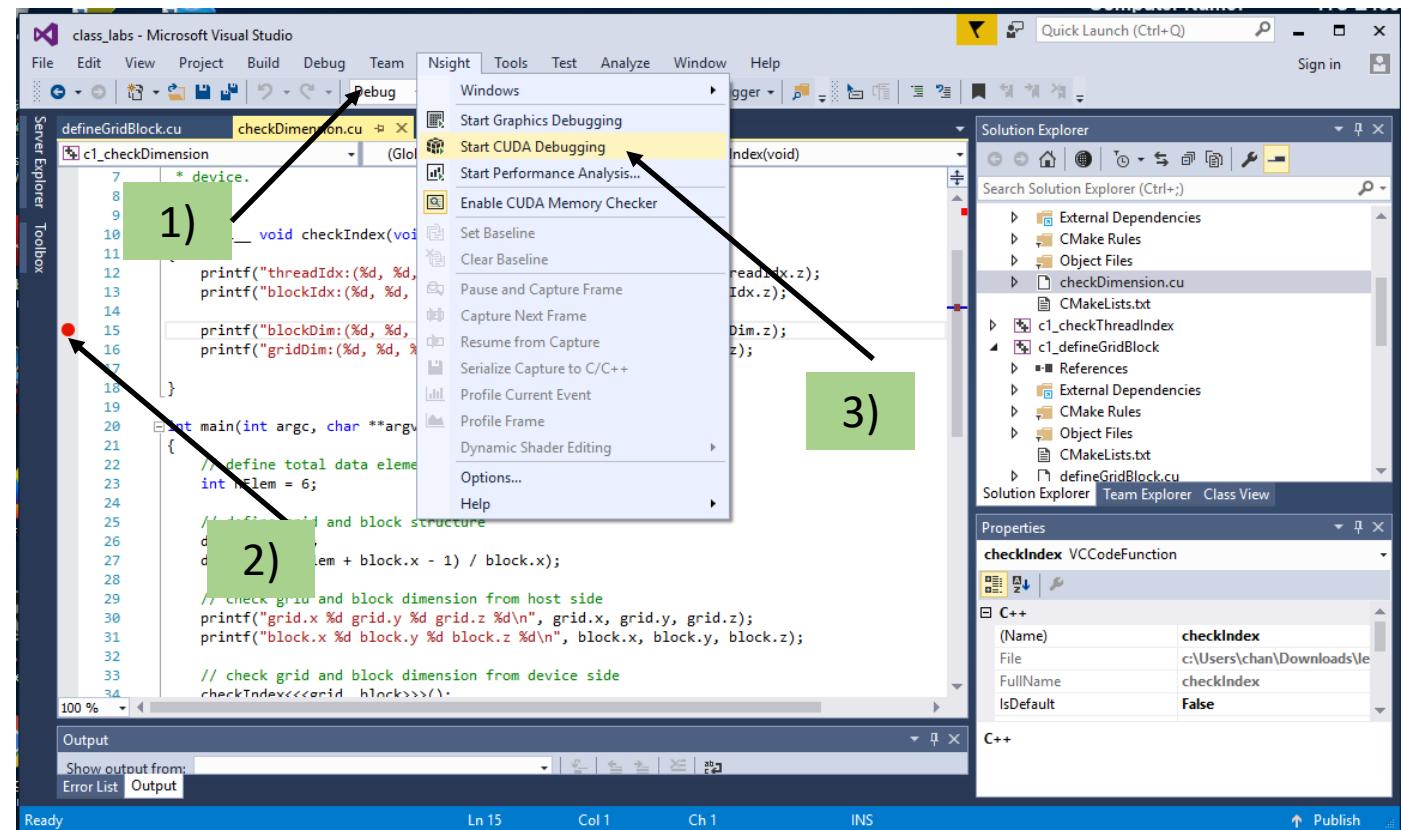
Address	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8
0x00000000C0000070	8.1999981	7.0000000	38.000000	26.399996	45.000000	21.600004	46.79999	
0x00000000C0000090	16.399996	28.799992	14.600004	48.200008	48.200008	37.4000015	46.59999	
0x00000000C00000B0	35.799992	33.200008	43.799992	12.000000	27.000000	2.4000010	12.39999	
0x00000000C00000F0	7.1999981	18.799992	2.5999990	5.5999990	1.2000005	36.599985	14.19999	
0x00000000C0000110	35.799992	3.5999990	15.399996	40.000000	13.399996	37.4000015	27.79999	
0x00000000C0000130	6.1999981	0.60000024	18.000000	25.000000	1.7999995	11.199998	7.40000	
0x00000000C0000150	18.600004	42.400015	40.599985	50.400015	30.000000	49.000000	13.80000	
0x00000000C0000170	3.7999995	2.5999990	27.399996	2.0000000	5.5999990	43.799992	34.79999	
0x00000000C0000190	6.4000010	30.799992	16.000000	47.599985	12.800002	24.000000	16.80000	
0x00000000C00001B0	3.5999990	14.600004	10.000000	49.200008	31.600004	25.000000	14.60000	
0x00000000C00001D0	34.599985	15.800002	4.0000000	48.400015	13.600004	12.800002	20.39999	
0x00000000C00001E0	21.399996	39.200008	9.6000038	36.599985	10.000000	11.800002	32.20000	

# Nsight VS Debug: GPU Kernel (1/3)

- Nsight VS debug is integrated into Visual Studio, but you should invoke “Nsight->Start CUDA Debug”
- CPU and GPU kernel debug cannot run simultaneously
  - Only one (CPU debug or GPU kernel debug) at a time

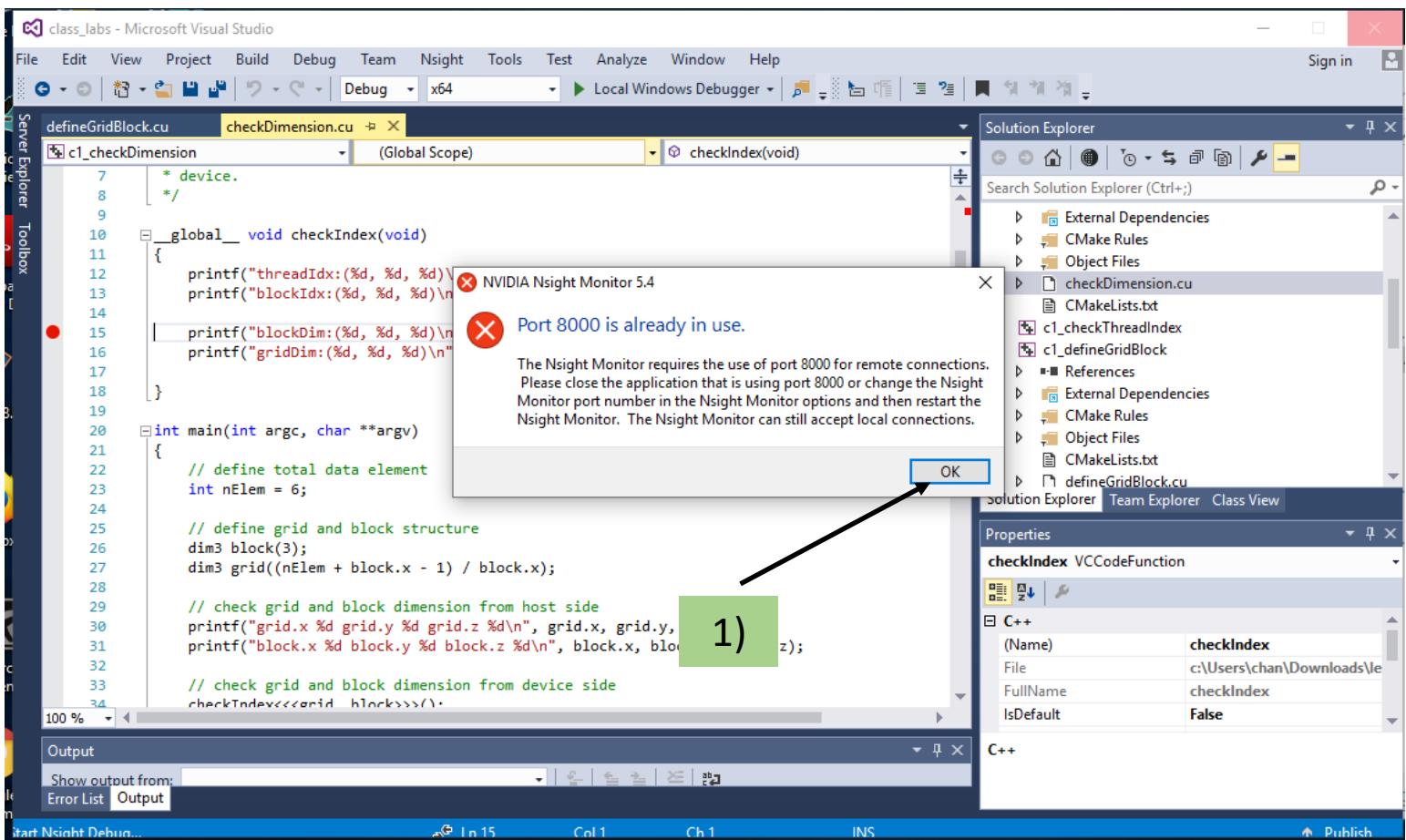
- 1) Set “Debug” mode
- 2) Add break points to CUDA kernels
- 3) “Start CUDA Debugging”

~~(legacy)~~



# Nsight VS Debug: GPU Kernel (2/3)

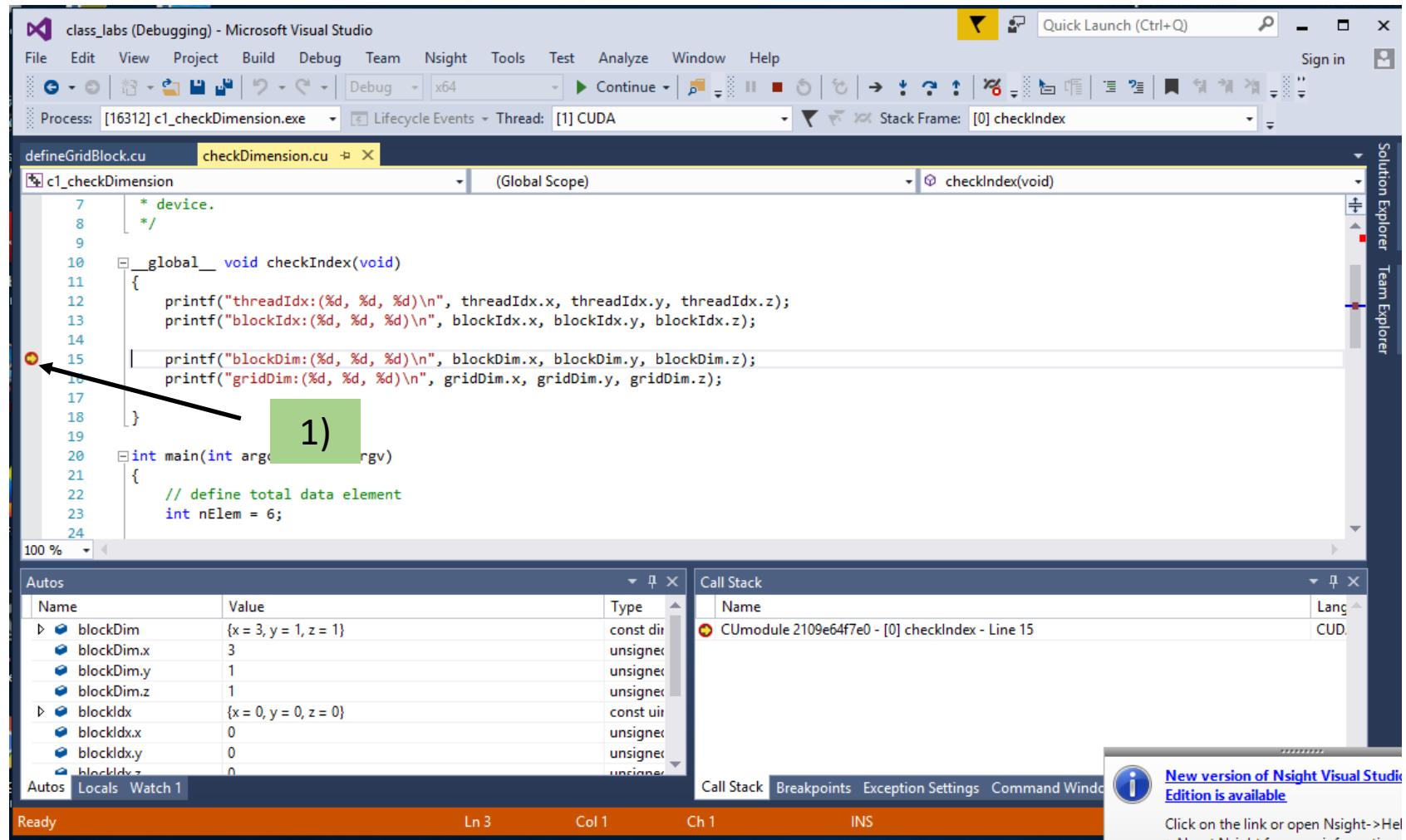
- 1) Just Click “OK” if you see  
“Port 8000 is already in  
use”



# Nsight VS Debug: GPU Kernel (3/3)

- 1) Confirm a “yellow arrow” when a kernel program stop at a break point

Watch 1 on  $\text{checkIndex}$   
variable of 2.  
not multiple threads  
debugging  
 $\Rightarrow$  특정 thread debug?  
(conditional breakpoint)



class\_labs (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Nsight Tools Test Analyze Window Help

Process: [16312] c1\_checkDimension.exe Lifecycle Events Thread: [1] CUDA Stack Frame: [0] checkIndex

defineGridBlock.cu checkDimension.cu

c1\_checkDimension (Global Scope) checkIndex(void)

```
7 * device.
8 */
9
10 __global__ void checkIndex(void)
11 {
12     printf("threadIdx:(%d, %d, %d)\n", threadIdx.x, threadIdx.y, threadIdx.z);
13     printf("blockIdx:(%d, %d, %d)\n", blockIdx.x, blockIdx.y, blockIdx.z);
14
15     printf("blockDim:(%d, %d, %d)\n", blockDim.x, blockDim.y, blockDim.z);
16     printf("gridDim:(%d, %d, %d)\n", gridDim.x, gridDim.y, gridDim.z);
17 }
18
19
20 int main(int argc, char* argv)
21 {
22     // define total data element
23     int nElem = 6;
24 }
```

Autos

Name	Value	Type
blockDim	{x = 3, y = 1, z = 1}	const dir
blockDim.x	3	unsigned
blockDim.y	1	unsigned
blockDim.z	1	unsigned
blockIdx	{x = 0, y = 0, z = 0}	const dir
blockIdx.x	0	unsigned
blockIdx.y	0	unsigned
blockIdx.z	0	unsigned

Call Stack

Name
CUmodule 2109e64f7e0 - [0] checkIndex - Line 15

Call Stack Breakpoints Exception Settings Command Window

New version of Nsight Visual Studio Edition is available

Click on the link or open Nsight->Help > About Nsight for more information

Ready Ln 3 Col 1 Ch 1 INS

# CUDA debugging Tips

- 1) Include Helper functions

```
#include <helper_functions.h>  
#include <helper_cuda.h>
```

- 2) Always add checkCudaErrors after calling cuda functions

```
checkCudaErrors(cudaMalloc());
```

breakpoint 를 걸어서 condition.  
운영체계에서

- ~~3) Always use a conditional break point for a specific thread (blockIdx, threadIdx)~~

```
@blockIdx(1,1,0) && @threadIdx(1,0,0)
```

⇒ cross-mark로 표기함.

- 4) Enable “Cuda-Memcheck” to check Device memory access violations

Night 템 밤에 있음.

- 5) Get error reports (refer to the next slide)

ⓐ 메모리에서

executed 된거나, 되고 있는 것 (번역선) 만 볼 수 있다.

# Conditional breakpoints

- @blockIdx(0,1,0) && @threadIdx(0,1,0)

Macro	Expansion	Notes
@threadIdx(x, y, z)	(threadIdx.x == (x) && threadIdx.y == (y) && threadIdx.z == (z))	x, y, and z must be decimal integers.
@threadIdx(#N)	( (((threadIdx.z * blockDim.y) + threadIdx.y) * blockDim.x + threadIdx.x) == (N) )	N must be a decimal integer.
@blockIdx(x, y, z)	(blockIdx.x == (x) && blockIdx.y == (y) && blockIdx.z == (z))	x, y, and z must be decimal integers.
@blockIdx(#N)	( (((blockIdx.z * gridDim.y) + blockIdx.y) * gridDim.x + blockIdx.x) == (N) )	N must be a decimal integer.

# Reporting Errors

- All CUDA API calls return an error code (`cudaError_t`)
  - Error in the API call itself
  - OR
  - Error in an earlier asynchronous operation (e.g. kernel)
- Get the error code for the last error:  
`cudaError_t cudaGetLastError(void)`
- Get a string to describe the error:  
`char *cudaGetString(cudaError_t)  
printf("%s\n", cudaGetString(cudaGetLastError()));`  
⇒ print 의미ful information 얻을 수 있다.

# CUDA Debug memory or registers as CPU Debug

- **View memory**

Really useful for CUDA debugging since you can only debug one thread

You cannot access the local variables of the other threads

However, the Global memory is common to all threads so it is also updated by the other threads

You can also check values of the other threads through the Global memory

- View variables

**You can only check local variables of a current thread**

- Warp watch

(other thread's  
variable 을 볼 수 X)