CUDA OPTIMIZATION WITH NVIDIA NSIGHT™ VISUAL STUDIO EDITION

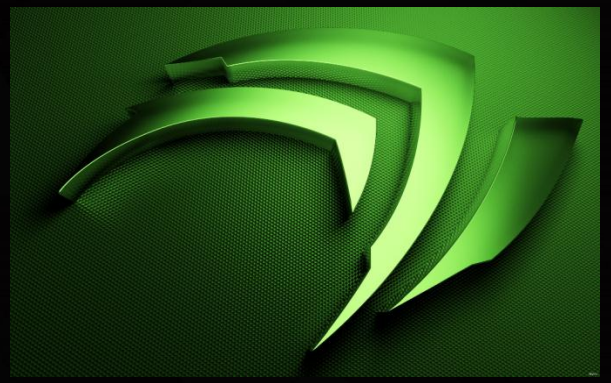
CHRISTOPH ANGERER, NVIDIA
JULIEN DEMOUTH, NVIDIA

WHAT YOU WILL LEARN

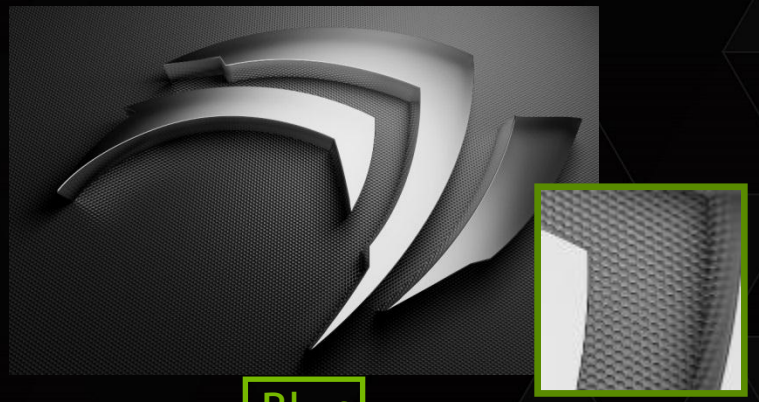
- An iterative method to optimize your GPU code
- A way to conduct that method with NVIDIA Nsight VSE
- Companion Code: https://github.com/chmaruni/nsight-gtc2015

GPU TECHNOLOGY CONFERENCE

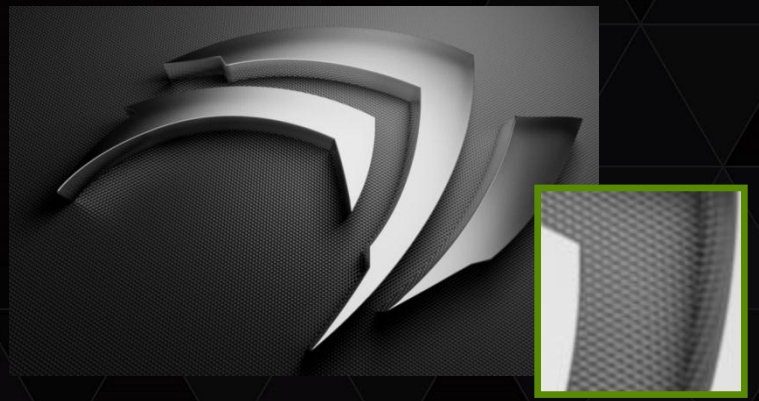
INTRODUCING THE APPLICATION



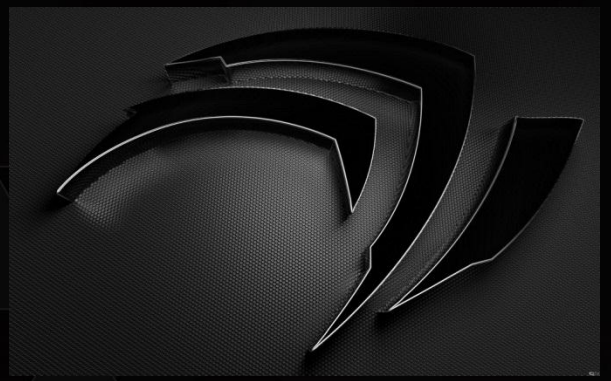
Grayscale



Blur



Edges



GPU TECHNOLOGY CONFERENCE

INTRODUCING THE APPLICATION

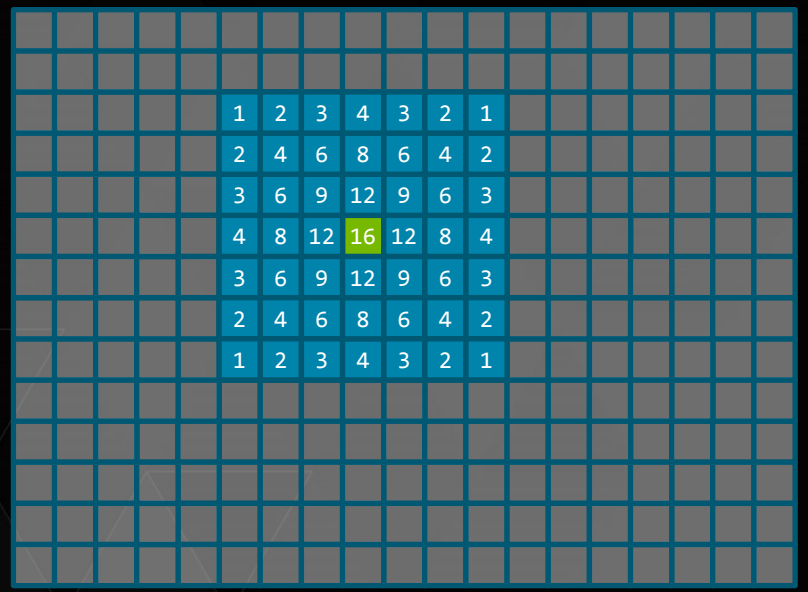
- Grayscale Conversion

```
// r, g, b: Red, green, blue components of the pixel p
foreach pixel p:
  p = 0.298839f*r + 0.586811f*g + 0.114350f*b;
```

GPU TECHNOLOGY CONFERENCE

INTRODUCING THE APPLICATION

- Blur: 7x7 Gaussian Filter




```
foreach pixel p:
  p = weighted sum of p and its 48 neighbors
```

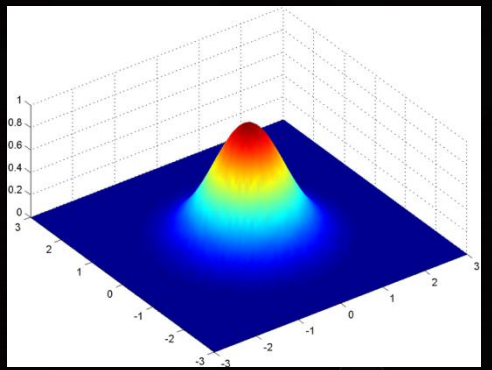
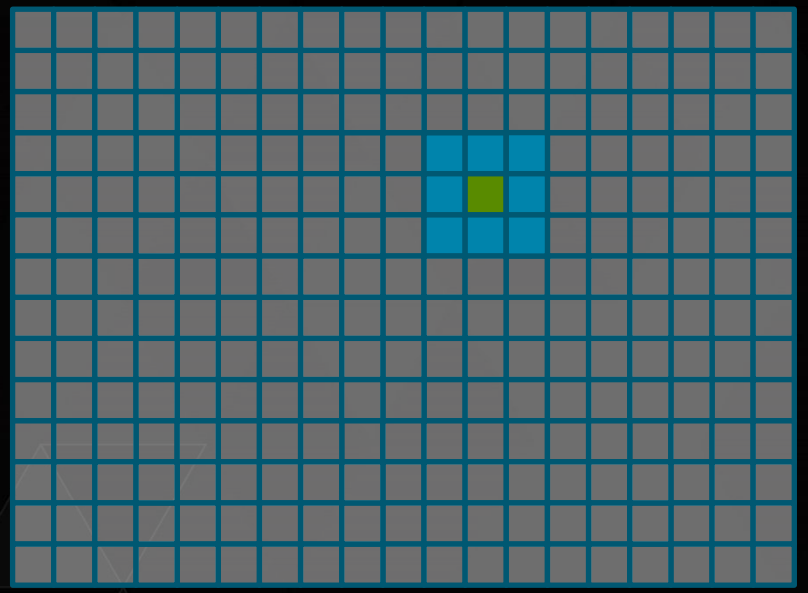


Image from Wikipedia

GPU TECHNOLOGY CONFERENCE

INTRODUCING THE APPLICATION

- Edges: 3x3 Sobel Filters



```
foreach pixel p:
  Gx = weighted sum of p and its 8 neighbors
  Gy = weighted sum of p and its 8 neighbors
  p  = sqrt(Gx + Gy)
```

Weights for Gx:

-1	0	1
-2	0	2
-1	0	1

Weights for Gy:

1	2	1
0	0	0
-1	-2	-1

GPU TECHNOLOGY CONFERENCE

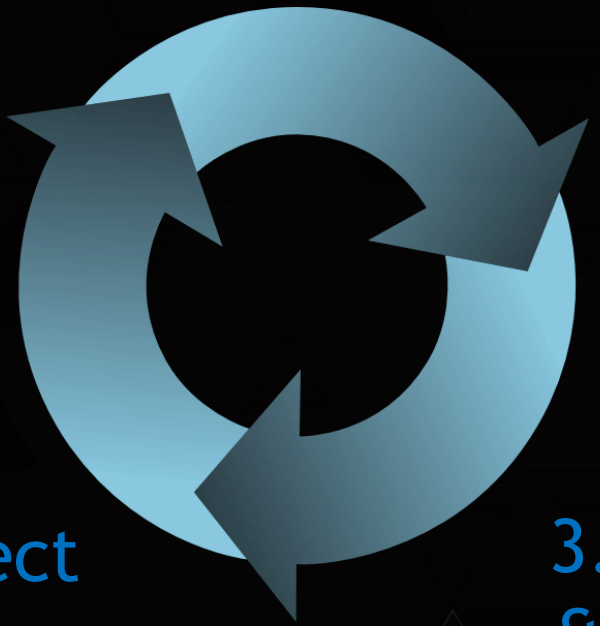
ENVIRONMENT

- NVIDIA GTX Titan X
  - GM200
  - SM5.2
- Windows 7
- NVIDIA Nsight Visual Studio Edition 4.6

GPU TECHNOLOGY CONFERENCE

PERFORMANCE OPTIMIZATION CYCLE

1. Profile Application



2. Identify Performance Limiter

3. Analyze Profile & Find Indicators

4. Reflect

4b. Build Knowledge

PREREQUISITES

- Basic understanding of the GPU Memory Hierarchy
  - Global Memory (slow, generous)
  - Shared Memory (fast, limited)
  - Registers (very fast, very limited)
  - (Texture Cache)
- Basic understanding of the CUDA execution model
  - Grid 1D/2D/3D
  - Block 1D/2D/3D
  - Warp-synchronous execution (32 threads per warp)

GPU TECHNOLOGY CONFERENCE

ITERATION 1

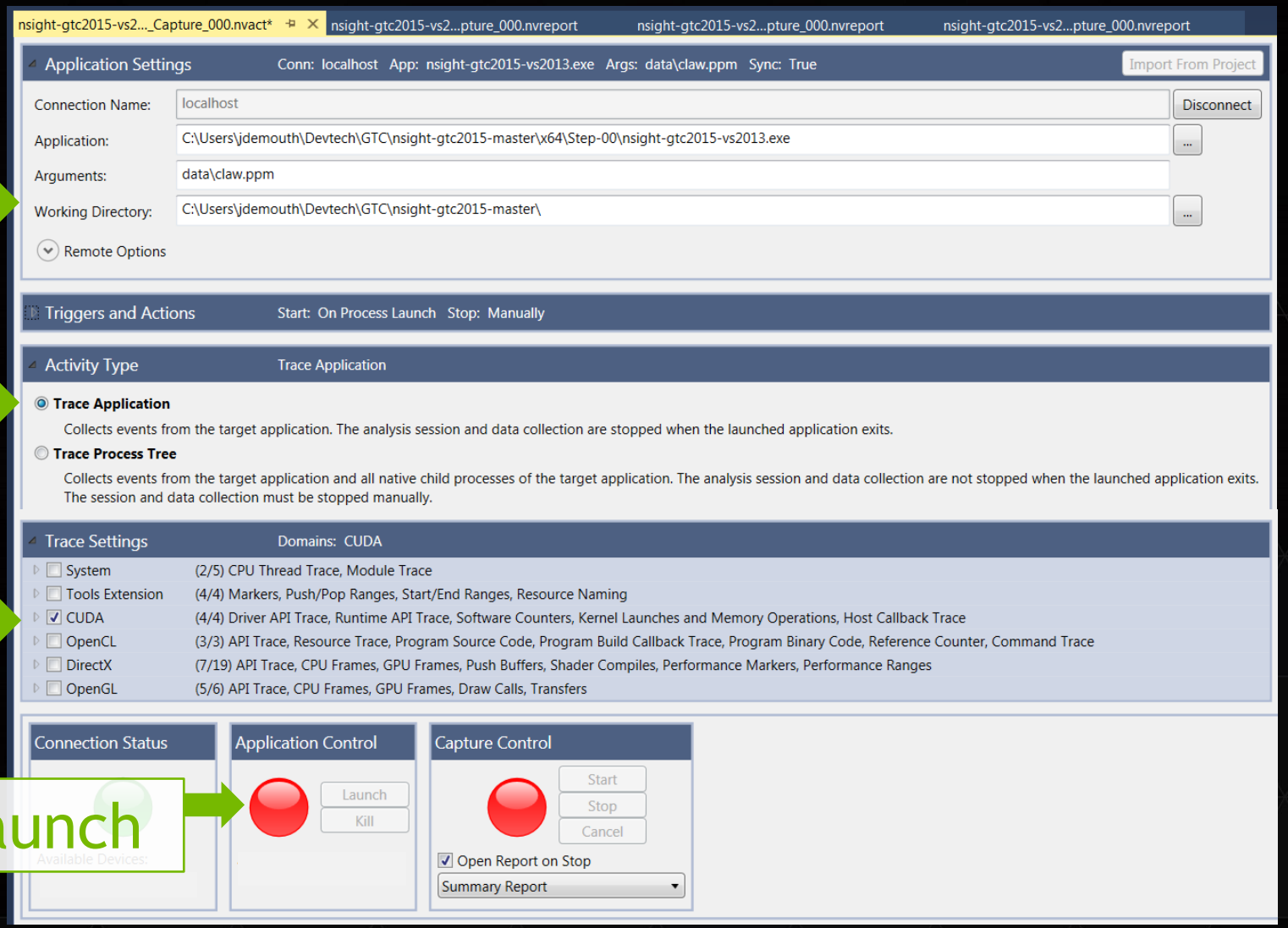
TRACING THE APPLICATION

Verify Parameters

Select Trace Application

Activate CUDA

Launch



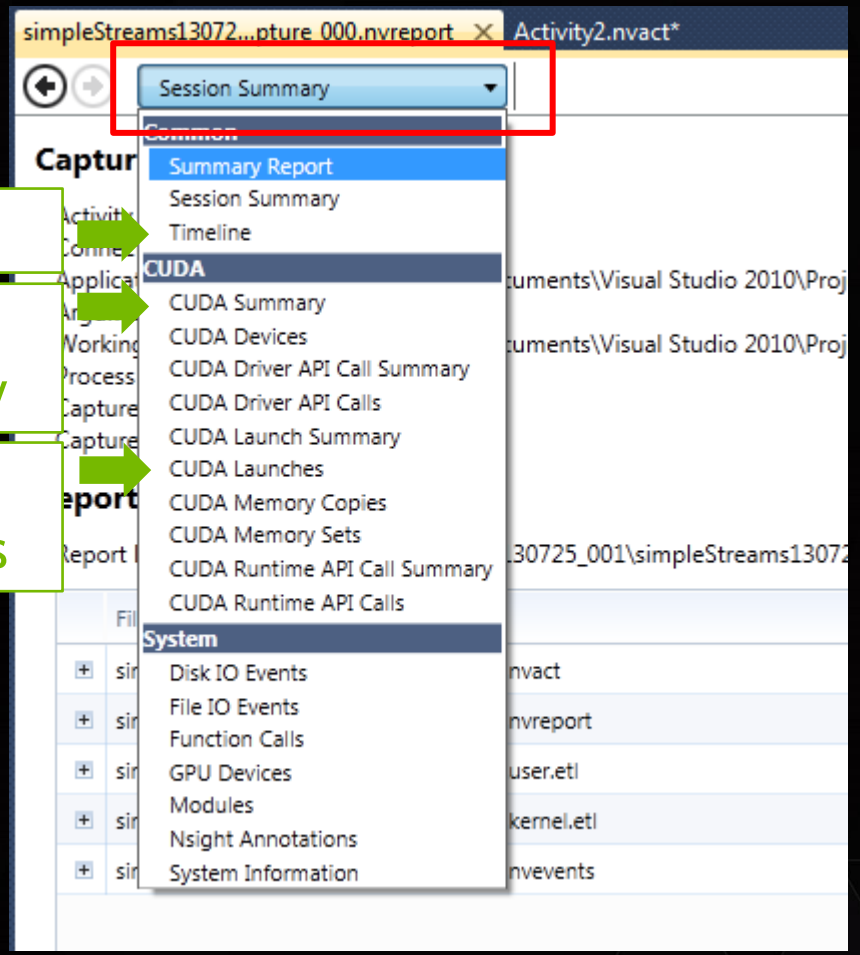
GPU TECHNOLOGY CONFERENCE

NAVIGATING THE ANALYSIS REPORTS

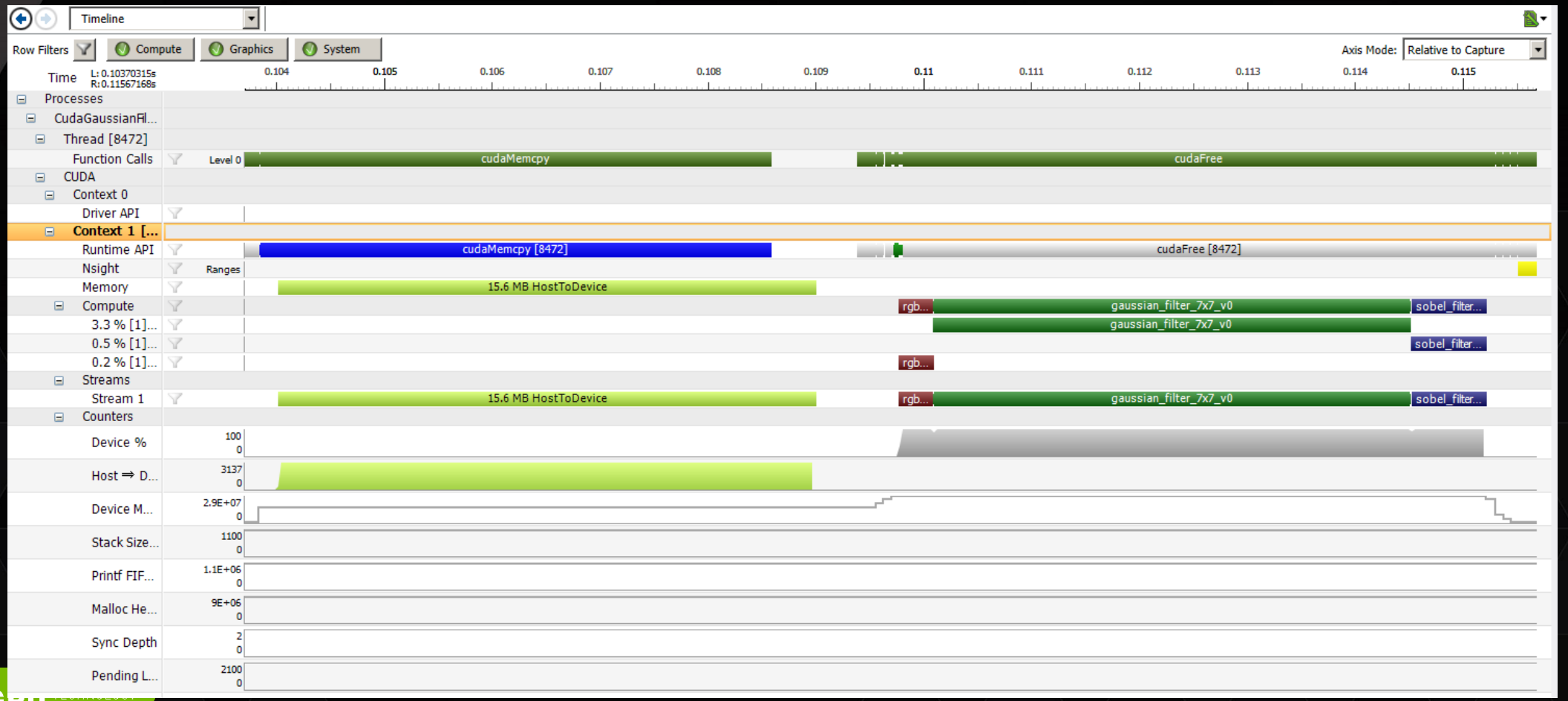
Timeline

CUDA Summary

CUDA Launches



TIMELINE



IDENTIFY HOTSPOT (CUDA SUMMARY)

Hotspot

Top Device Functions By Total Time Summary | All

	Name	Launches	Device %
1	gaussian_filter_7x7_v6	1	0.01
2	sobel_filter_3x3_v1	1	0.01
3	rgba_to_grayscale_kernel_v1	1	0.00

- Identify the hotspot: gaussian_filter_7x7_v0()

Kernel	Time	Speedup
Original Version	1.971ms	1.00x

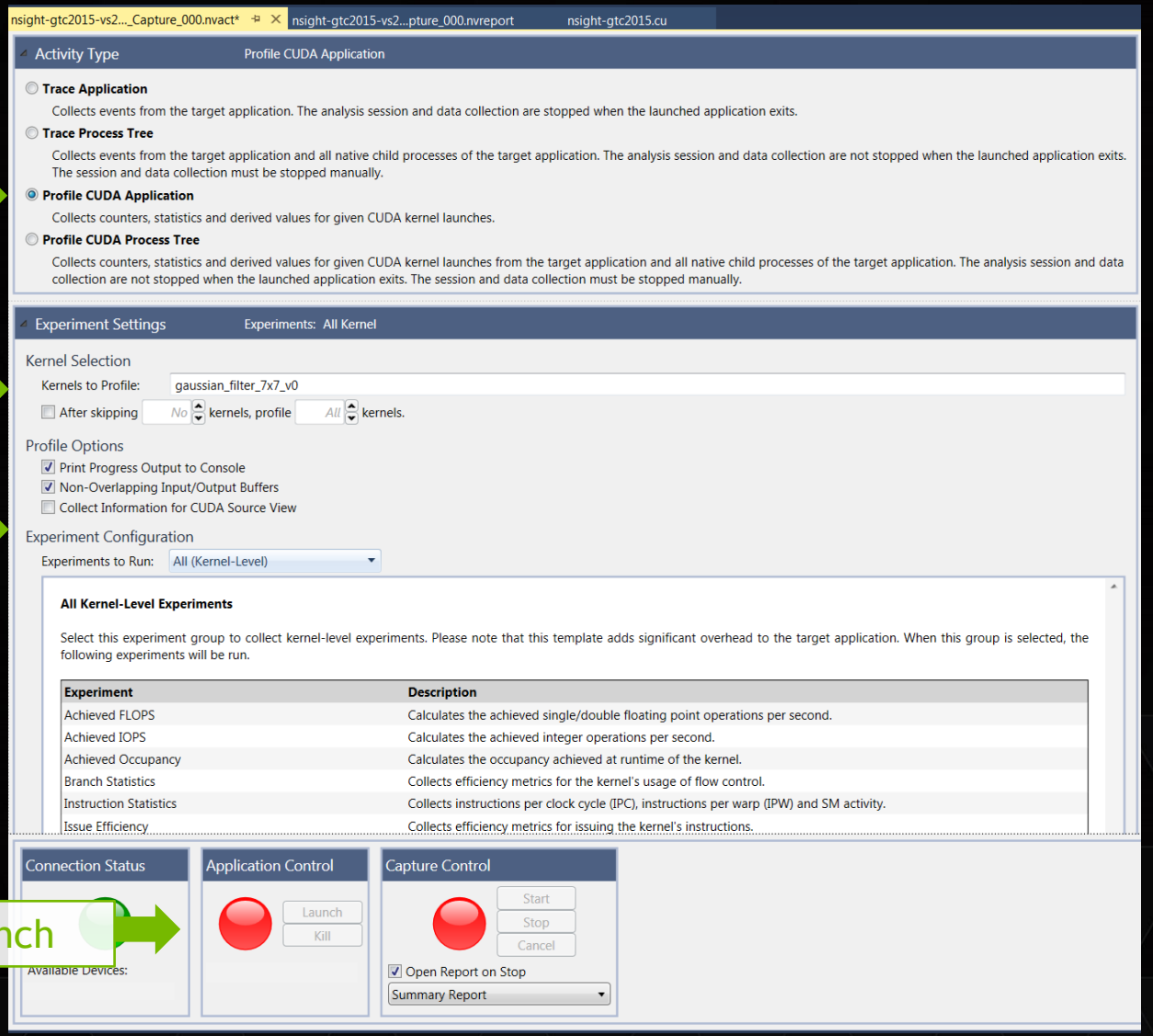
PERFORM KERNEL ANALYSIS

Select Profile CUDA Application

Select the Kernel

Select the Experiments (All)

Launch



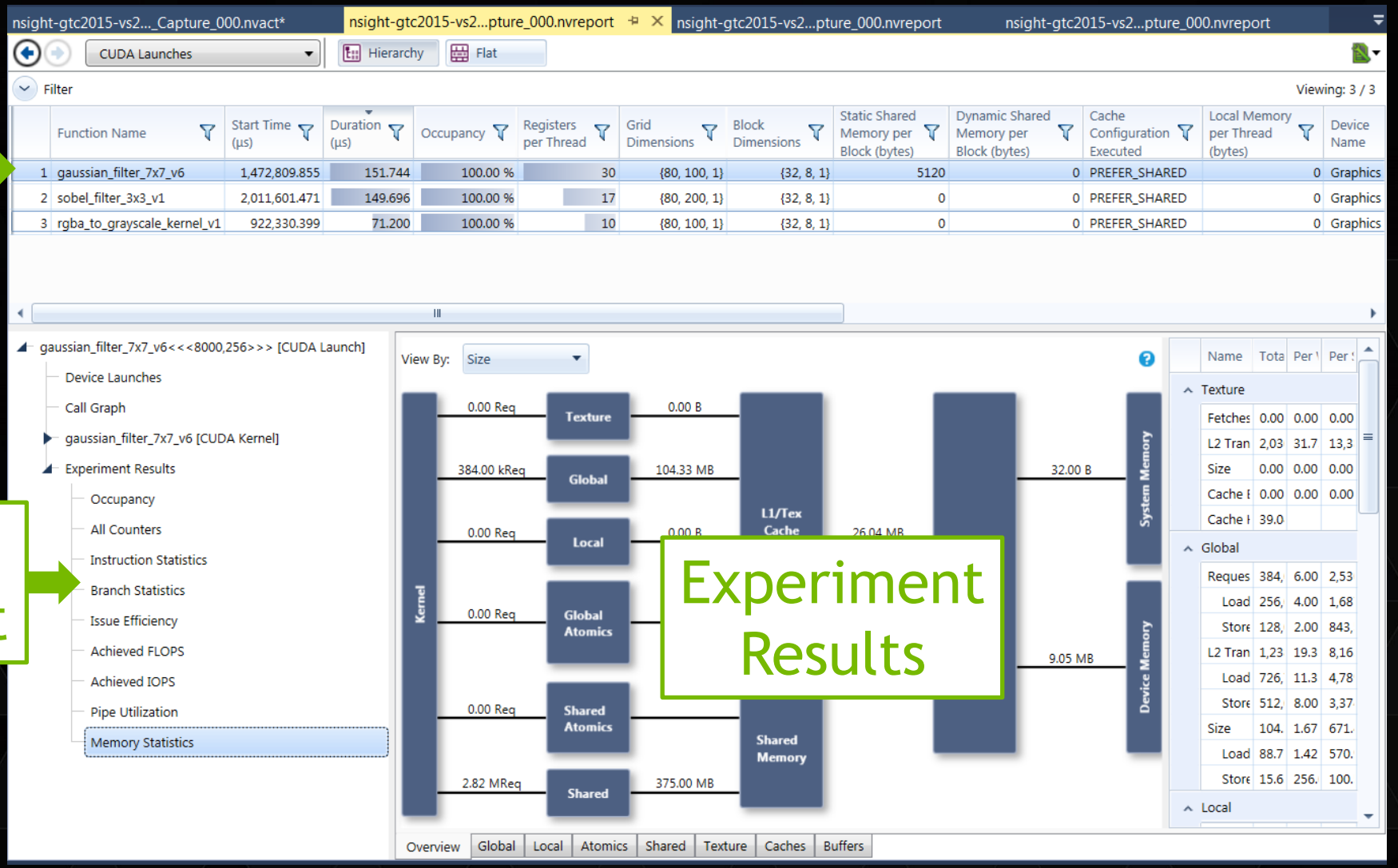
GPU TECHNOLOGY CONFERENCE

THE CUDA LAUNCHES VIEW

Select Kernel

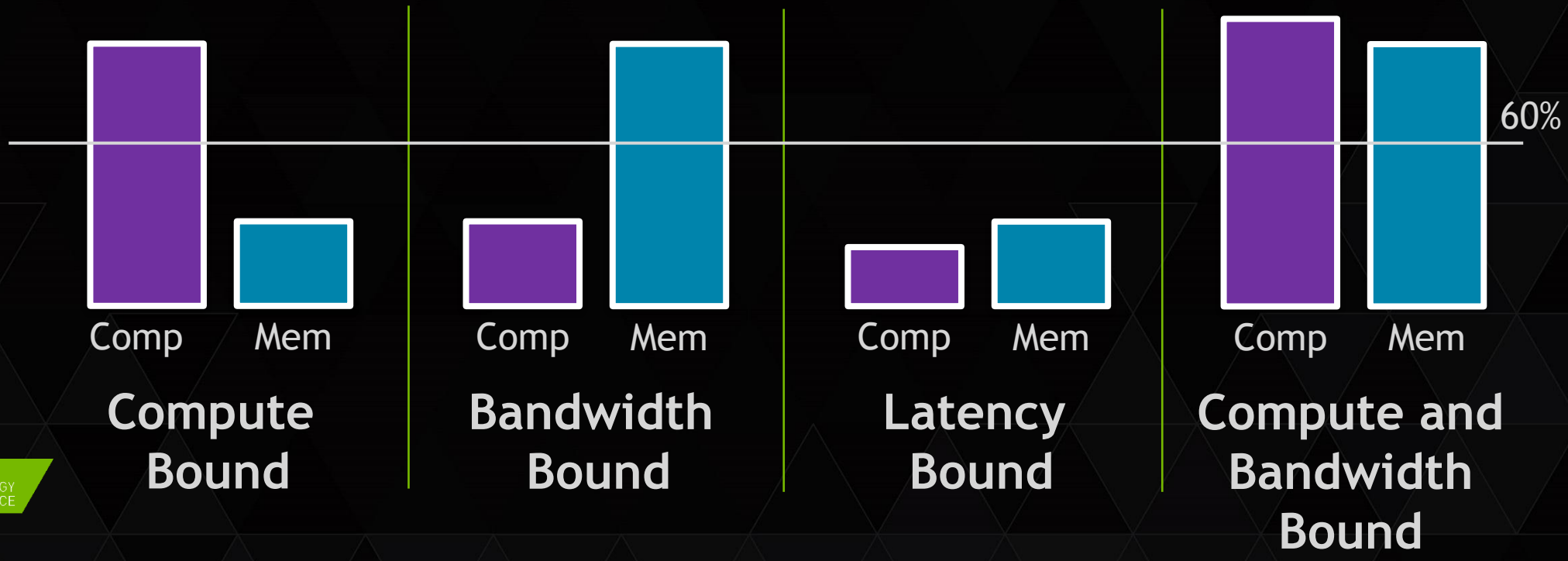
Select Experiment

Experiment Results

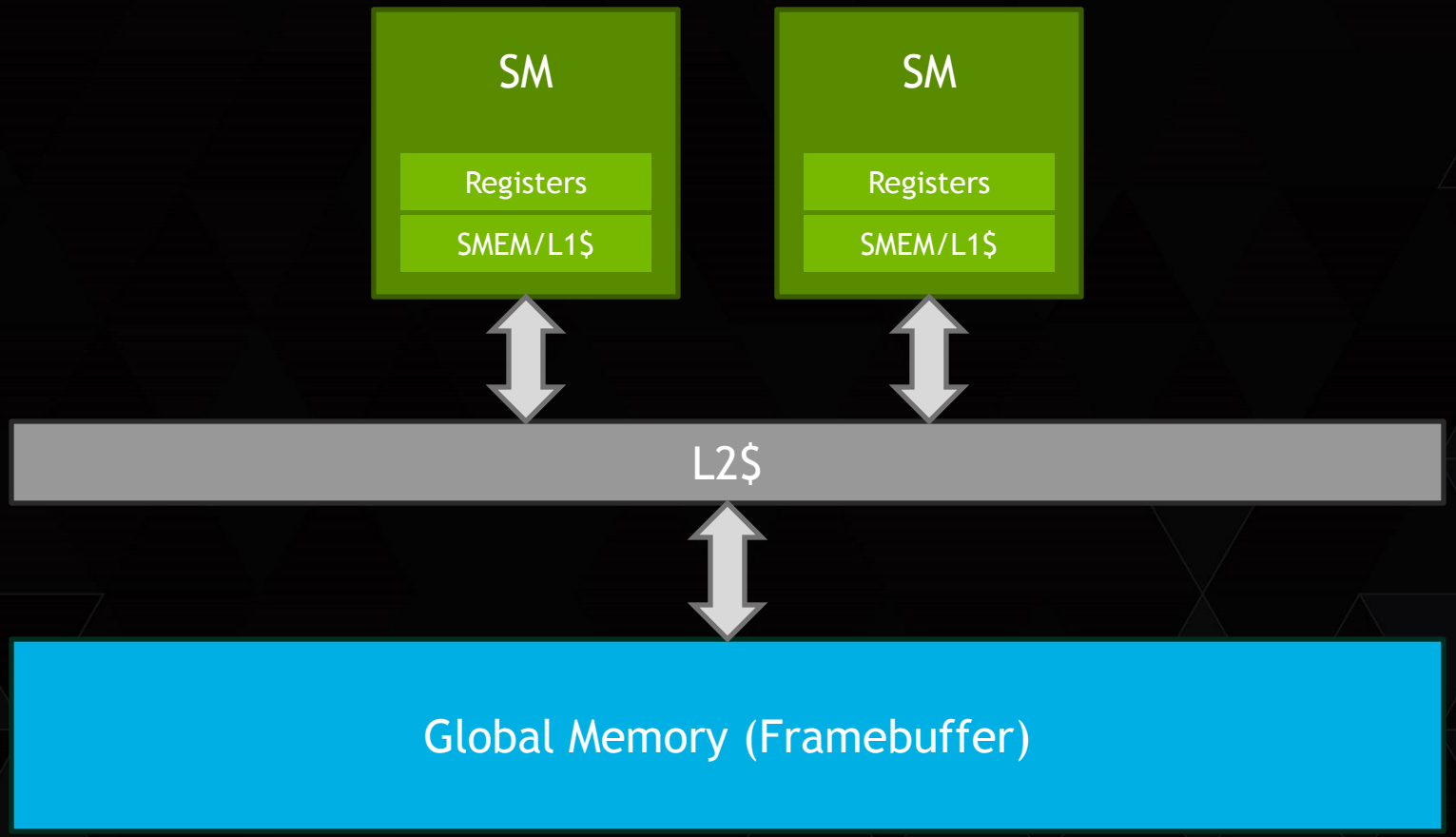


IDENTIFY MAIN PERFORMANCE LIMITER

- Memory Utilization vs Compute Utilization
- Four possible combinations:



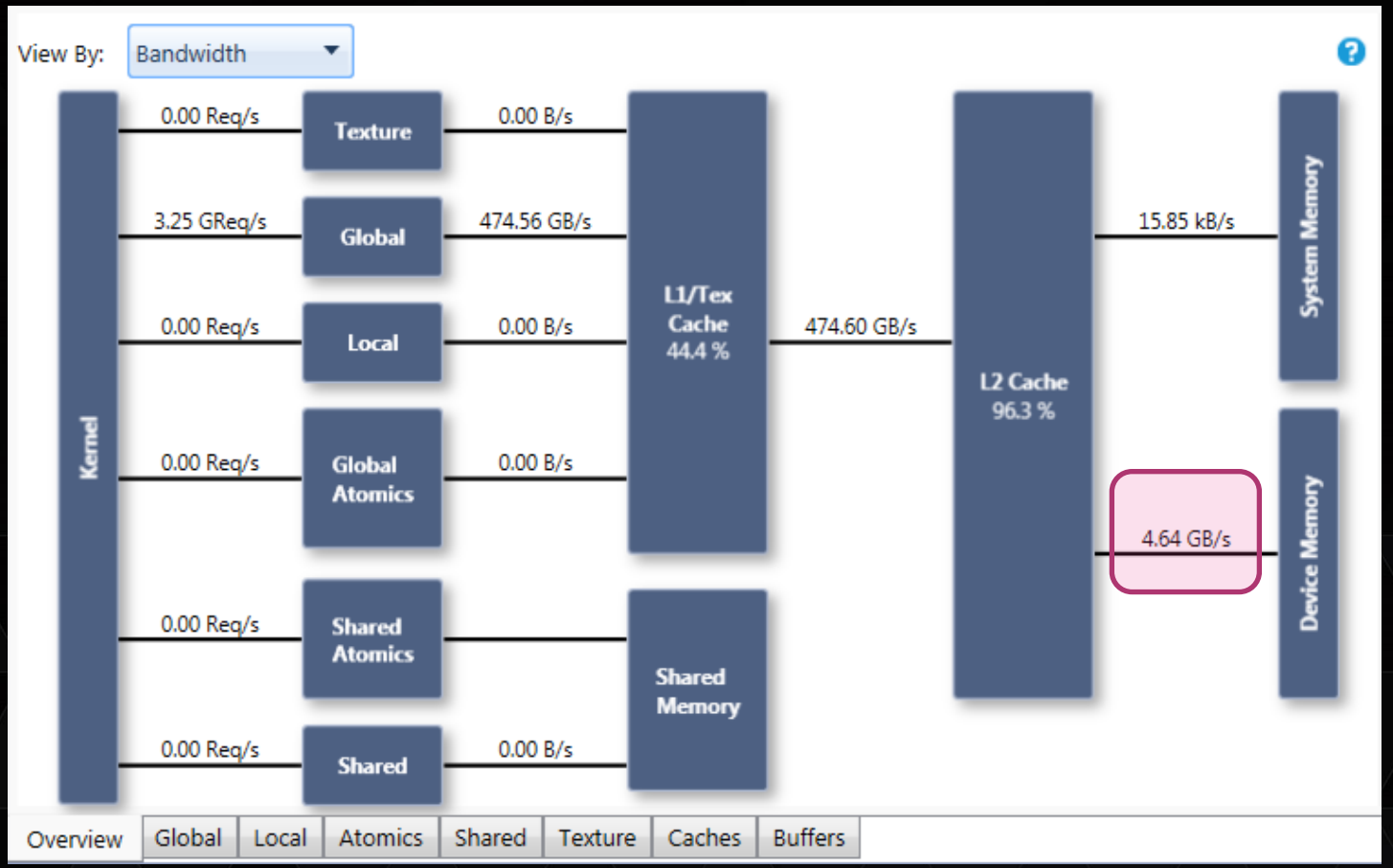
MEMORY BANDWIDTH



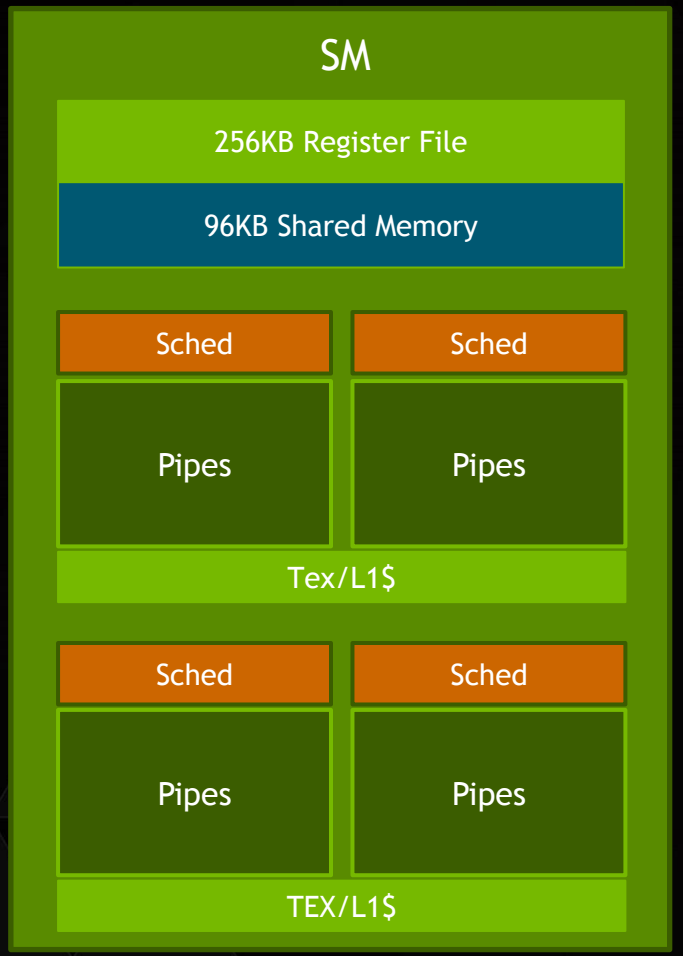
GPU TECHNOLOGY CONFERENCE

IDENTIFY PERFORMANCE LIMITER

- Utilization of L2$ Bandwidth (BW) limited and DRAM BW < 2%
- Not limited by memory bandwidth



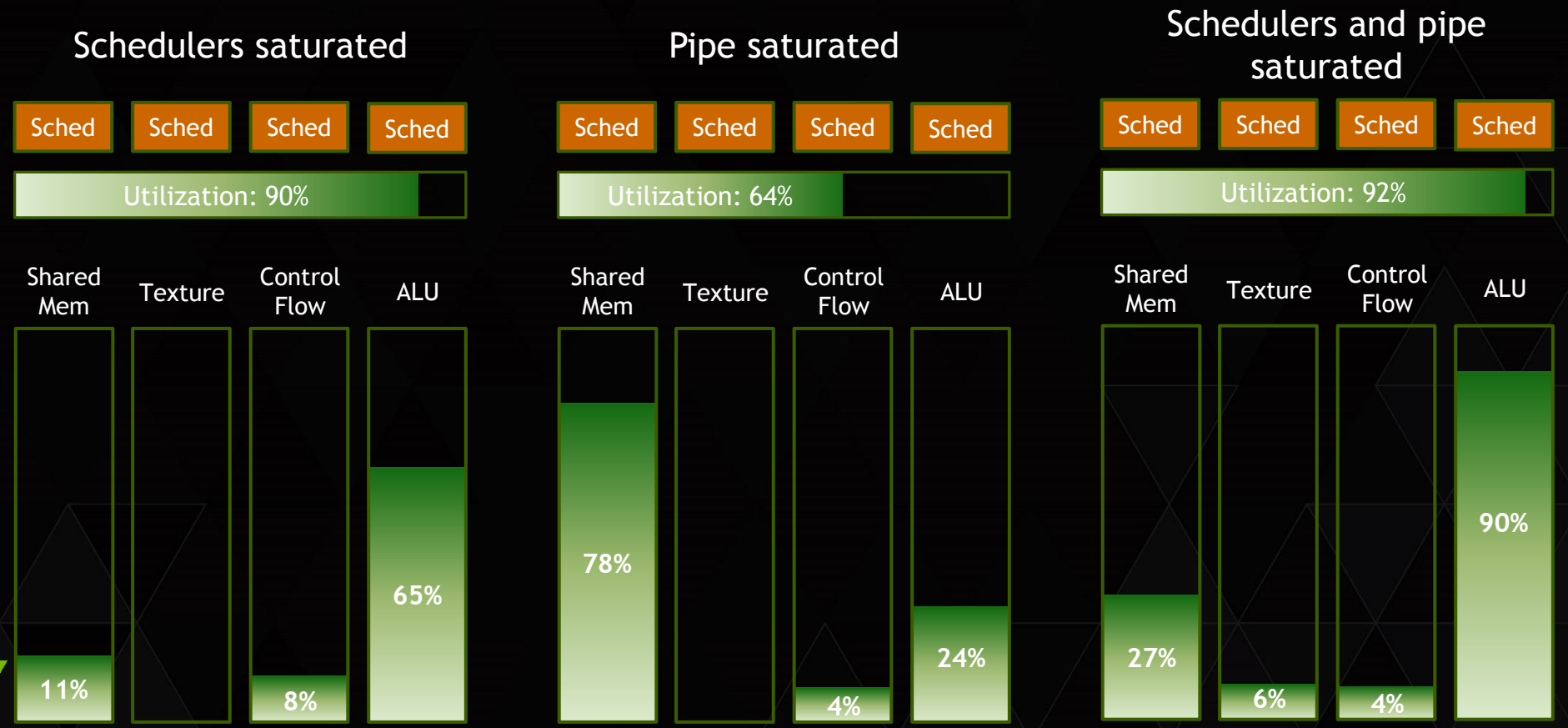
INSTRUCTION THROUGHPUT



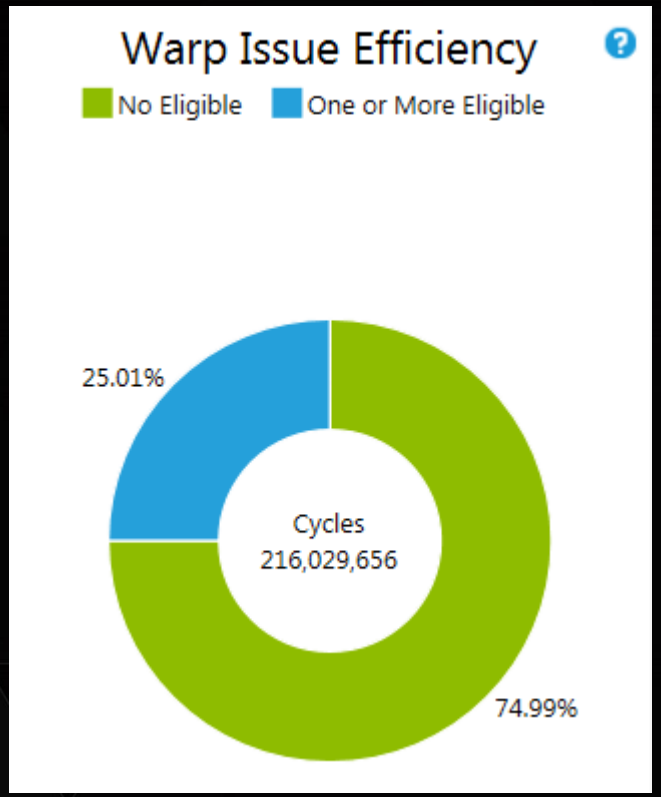
- Each SM has 4 schedulers (Maxwell)
- Schedulers issue instructions to pipes
- Each scheduler schedules up to 2 instructions per cycle
- A scheduler issues inst. from a single warp
- Cannot issue to a pipe if its issue slot is full

GPU TECHNOLOGY CONFERENCE

INSTRUCTION THROUGHPUT



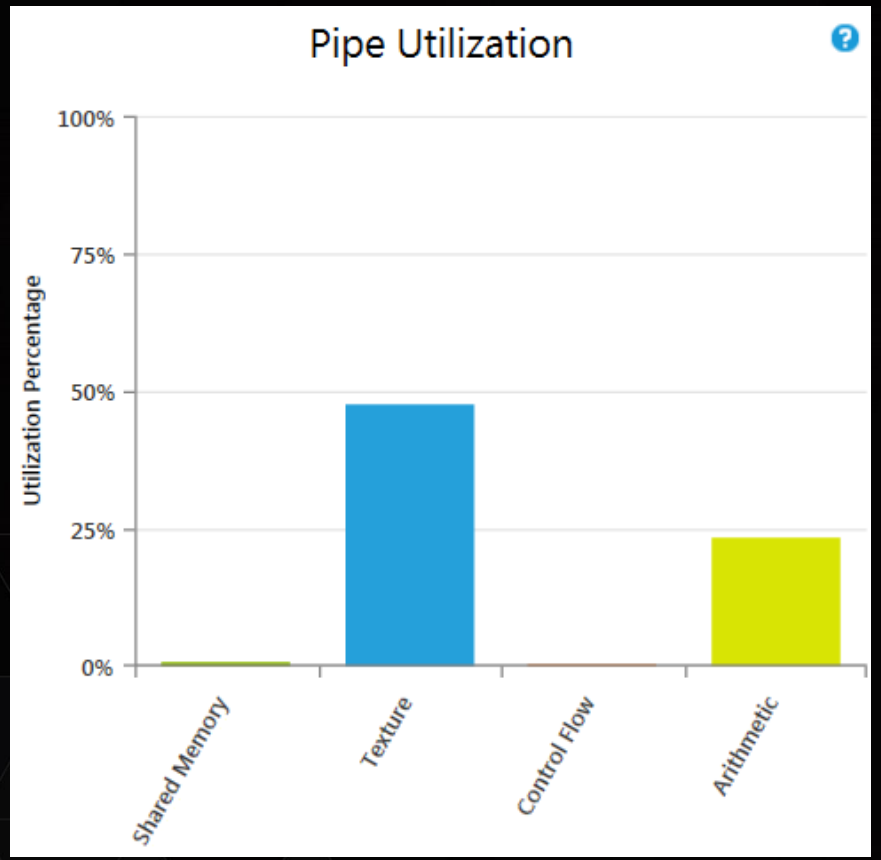
WARP ISSUE EFFICIENCY



- Percentage of issue slots used (blue)
- Aggregated over all the schedulers

GPU TECHNOLOGY CONFERENCE

PIPE UTILIZATION

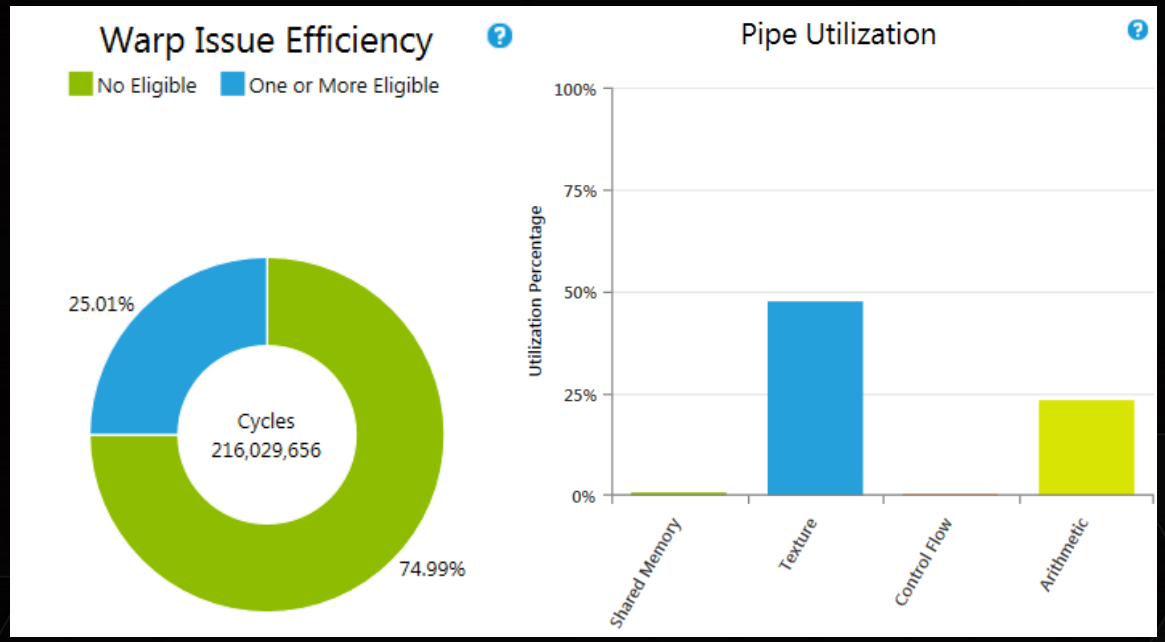


- Percentages of issue slots used per pipe
- Accounts for pipe throughputs
- Four groups of pipes:
  - Shared Memory
  - Texture
  - Control Flow
  - Arithmetic (ALU)

GPU TECHNOLOGY CONFERENCE

INSTRUCTION THROUGHPUT

- Neither schedulers nor pipes are saturated

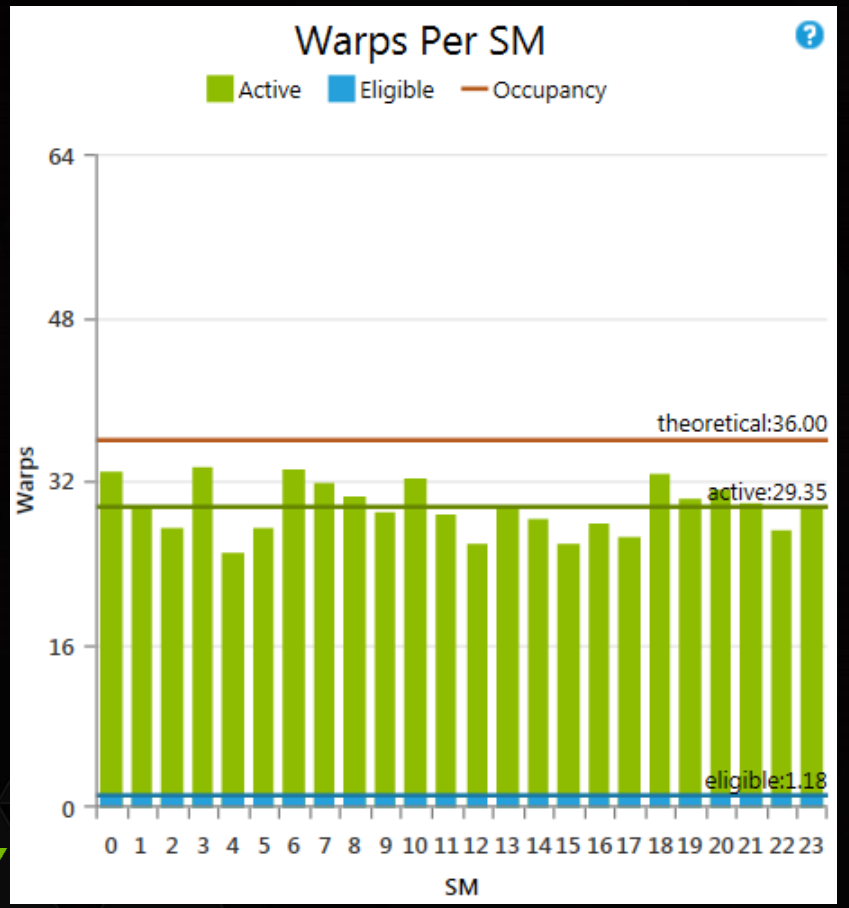


- Not limited by the instruction throughput

⇒ Our Kernel is Latency Bound

GPU TECHNOLOGY CONFERENCE

LOOKING FOR INDICATORS



- 56% of theoretical occupancy
- 29.35 active warps per cycle
- 1.18 warps eligible per cycle
- Let’s start with occupancy

GPU TECHNOLOGY CONFERENCE

OCCUPANCY

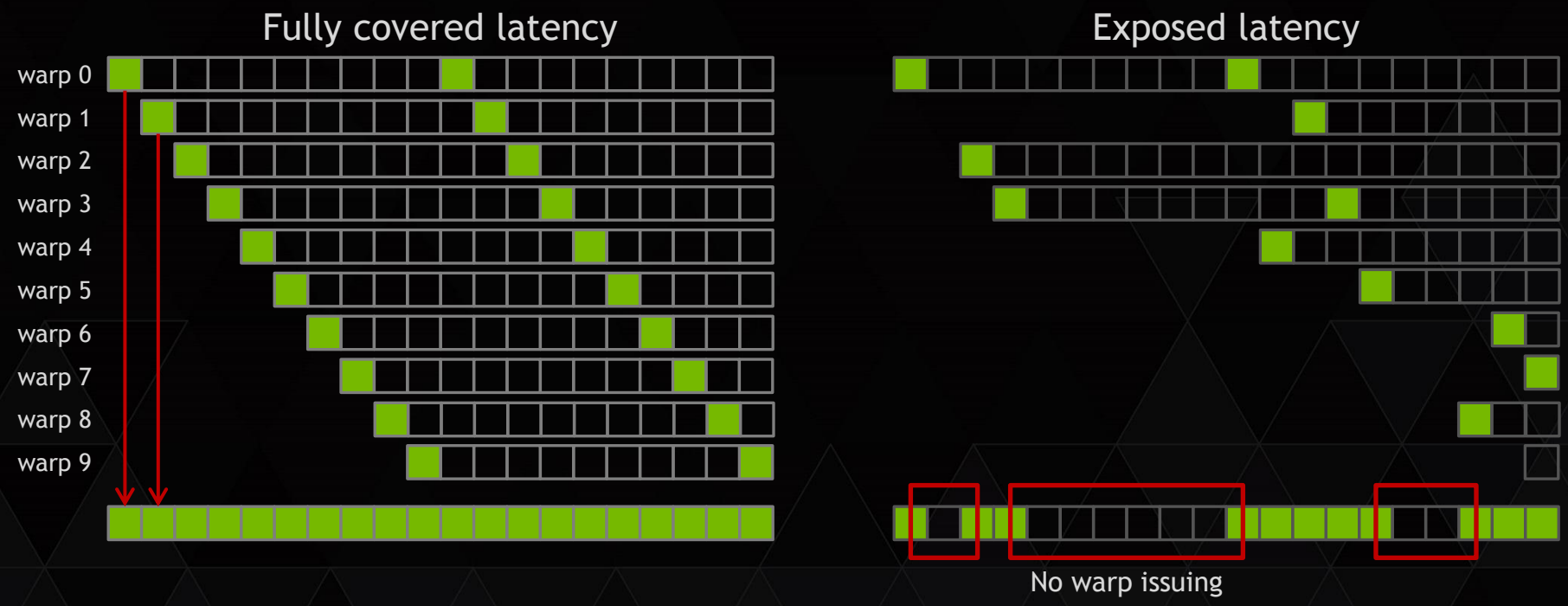
- Each SM has limited resources
- 64K Registers (32 bit) shared by threads
- Up to 48KB of shared memory per block (96KB per SMM)
- 32 Active Blocks per SMM
- Full occupancy: 2048 threads per SM (64 warps)

GPU TECHNOLOGY CONFERENCE

Values vary with Compute Capability

LATENCY

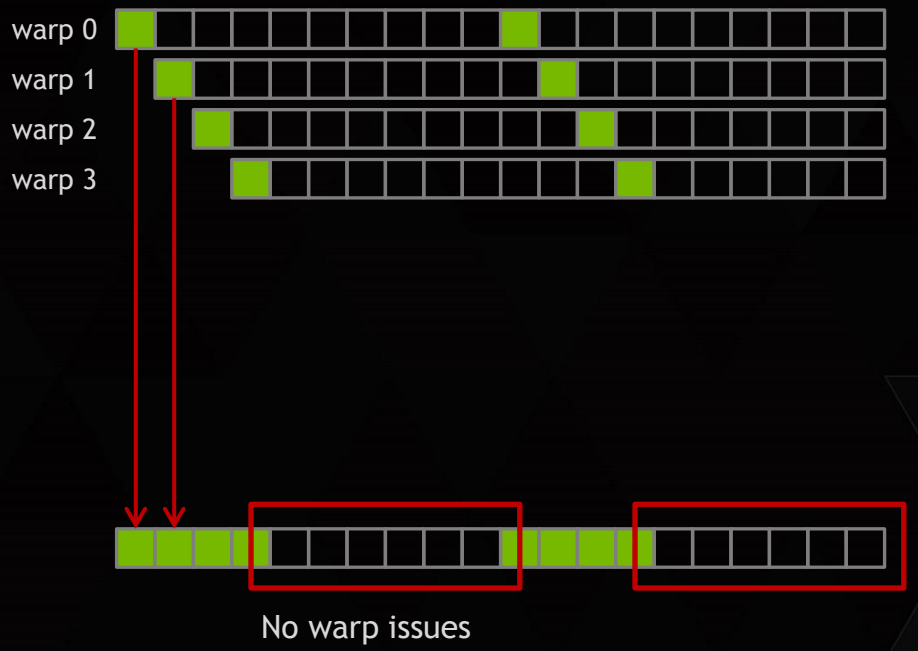
- GPUs cover latencies by having a lot of work in flight



GPU TECHNOLOGY CONFERENCE

LATENCY: LACK OF OCCUPANCY

- Not enough active warps



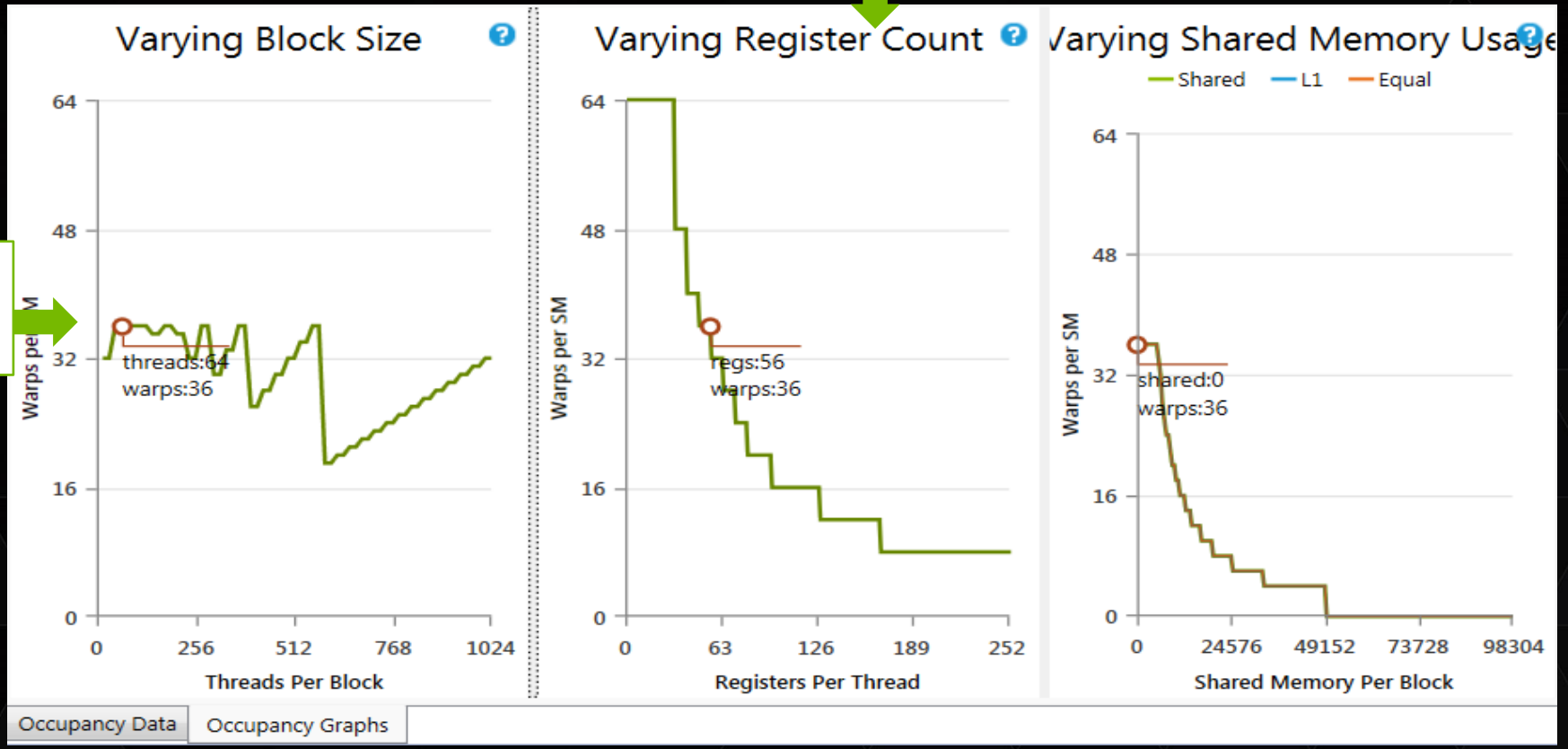
GPU TECHNOLOGY CONFERENCE

- The schedulers cannot find eligible warps at every cycle

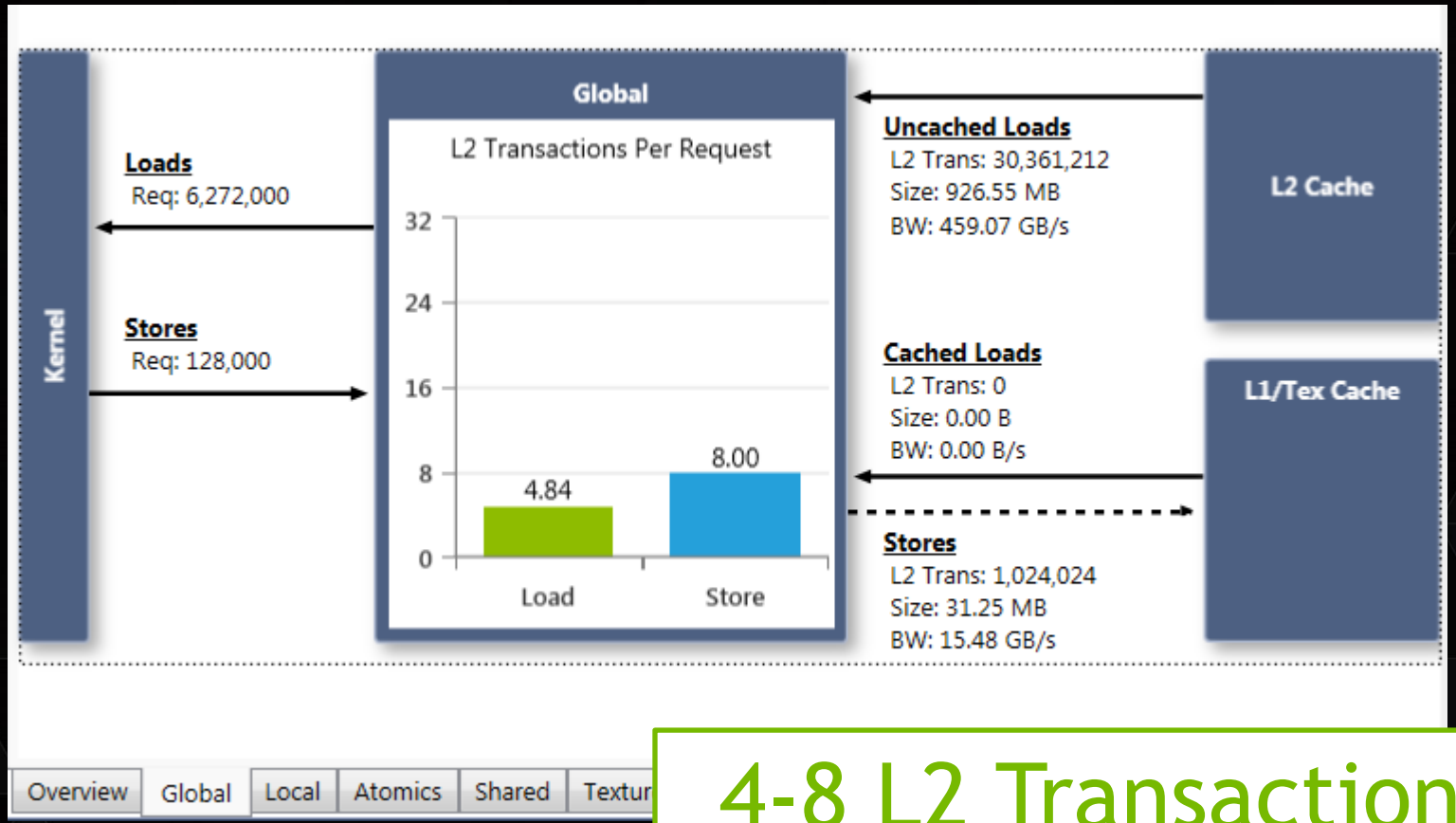
LOOKING FOR MORE INDICATORS

We don't want to change the register count yet

Block Size seems OK



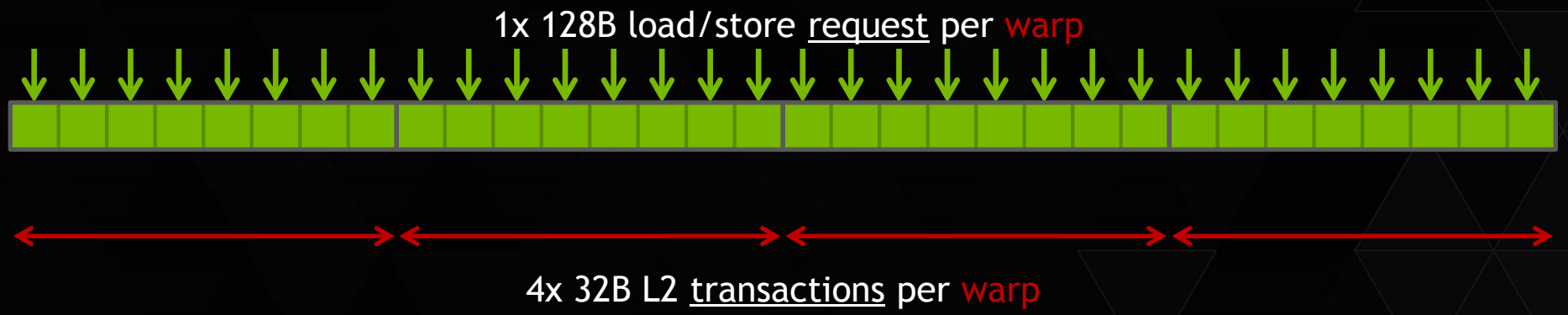
CONTINUE LOOKING FOR INDICATORS



4-8 L2 Transactions per 1 Request

MEMORY TRANSACTIONS: BEST CASE

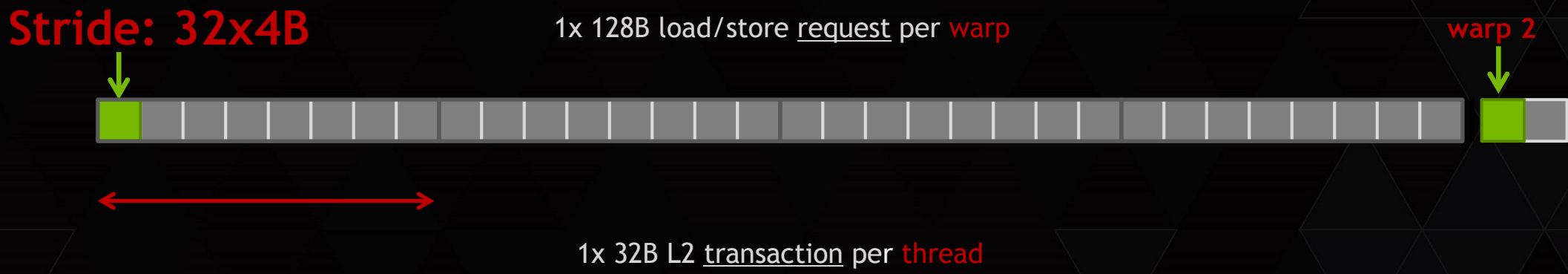
- A warp issues 32x4B aligned and consecutive load/store request
- Threads read different elements of the same 128B segment



- 4x L2 transactions: 128B needed / 128B transferred

MEMORY TRANSACTIONS: WORST CASE

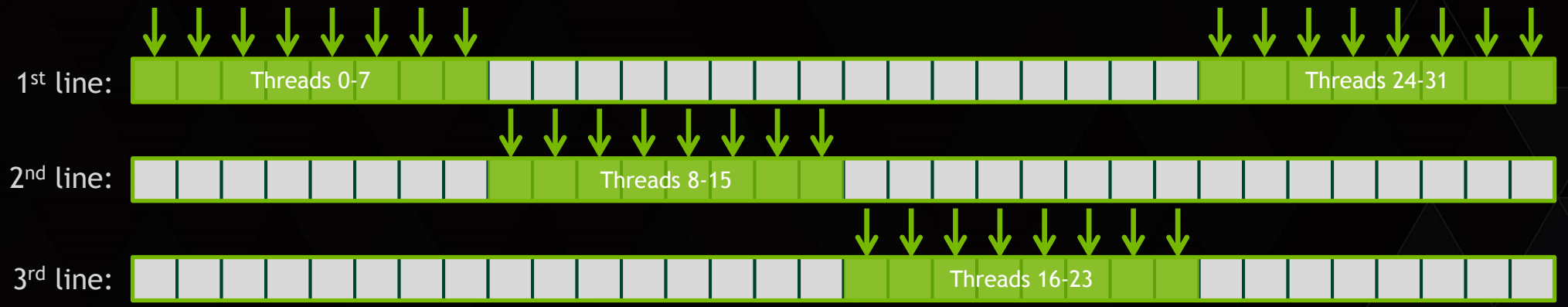
- Threads in a warp read/write 4B words, 128B between words
- Each thread reads the first 4B of a 128B segment



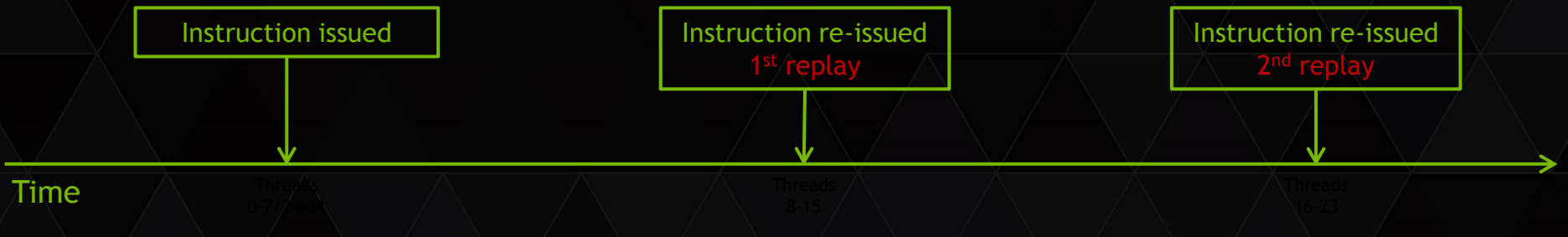
- 32x L2 transactions: 128B needed / 32x 32B transferred

TRANSACTIONS AND REPLAYS

- A warp reads from addresses spanning 3 lines of 128B

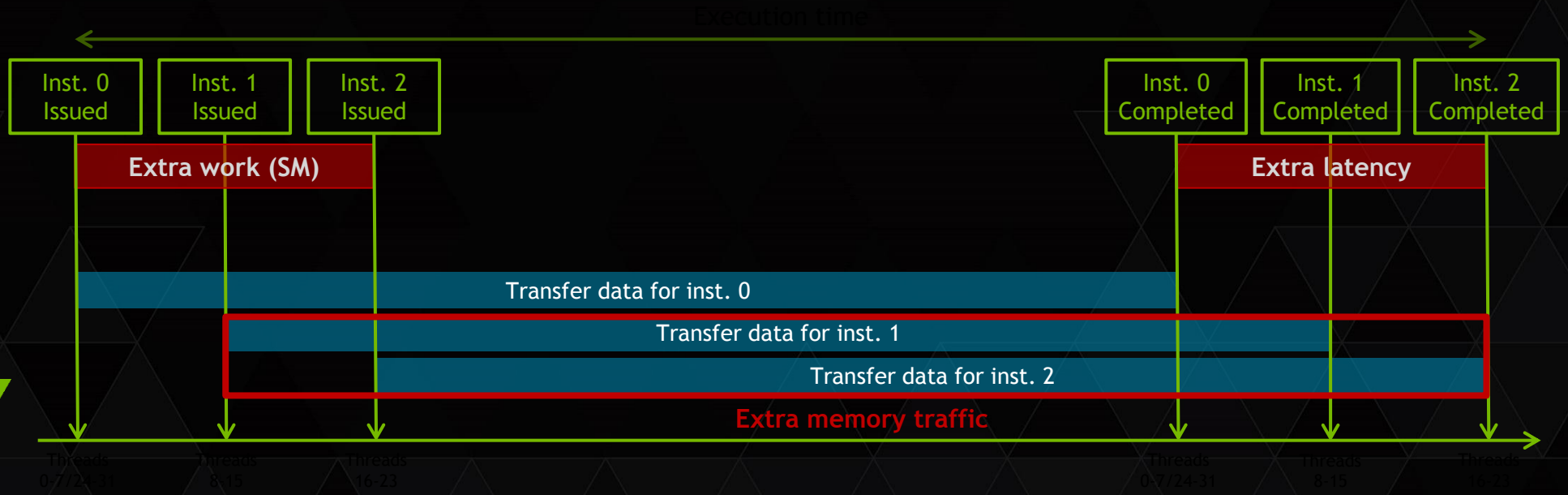


- 1 instr. executed and 2 replays = 1 request and 3 transactions



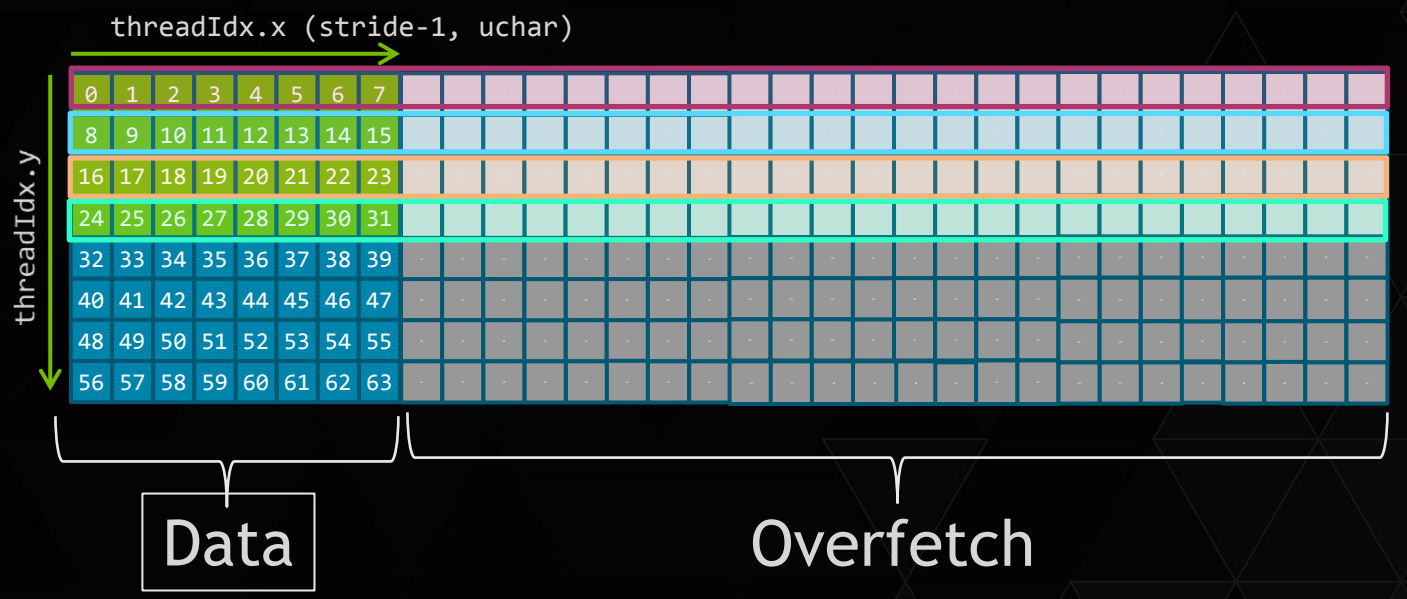
TRANSACTIONS AND REPLAYS

- With replays, requests take more time and use more resources
  - More instructions issued
  - More memory traffic
  - Increased execution time

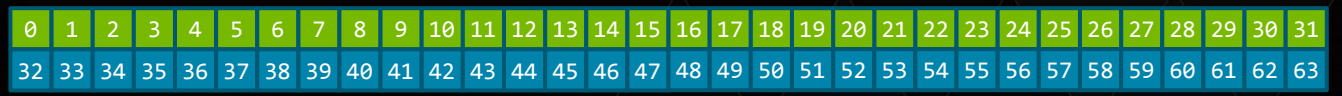


CHANGING THE BLOCK LAYOUT

- Our blocks are 8x8



- We should use blocks of size 32x2



GPU TECHNOLOGY CONFERENCE

IMPROVED MEMORY ACCESS

- Blocks of size 32x2
- Memory is used more efficiently

Kernel	Time	Speedup
Original Version	1.971ms	1.00x
Better Memory Accesses	0.725ms	2.72x

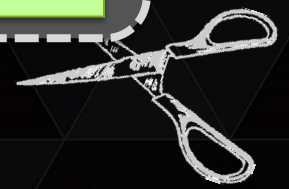
GPU TECHNOLOGY CONFERENCE

PERF-OPT QUICK REFERENCE CARD

Category:	Latency Bound - Occupancy
Problem:	Latency is exposed due to low occupancy
Goal:	**Hide** latency behind more parallel work
Indicators:	Occupancy low (< 60%) Execution Dependency High
Strategy:	Increase occupancy by: • Varying block size • Varying shared memory usage • Varying register count

PERF-OPT QUICK REFERENCE CARD

Category:	Latency Bound - Coalescing
Problem:	Memory is accessed inefficiently => high latency
Goal:	Reduce #transactions/request to reduce latency
Indicators:	Low global load/store efficiency, High #transactions/#request compared to ideal
Strategy:	Improve memory coalescing by: • Cooperative loading inside a block • Change block layout • Aligning data • Changing data layout to improve locality



PERF-OPT QUICK REFERENCE CARD

Category:	Bandwidth Bound - Coalescing
Problem:	Too much unused data clogging memory system
Goal:	Reduce traffic, move more useful data per request
Indicators:	Low global load/store efficiency, High #transactions/#request compared to ideal
Strategy:	Improve memory coalescing by: • Cooperative loading inside a block • Change block layout • Aligning data • Changing data layout to improve locality

ITERATION 2

IDENTIFY HOTSPOT

Hotspot

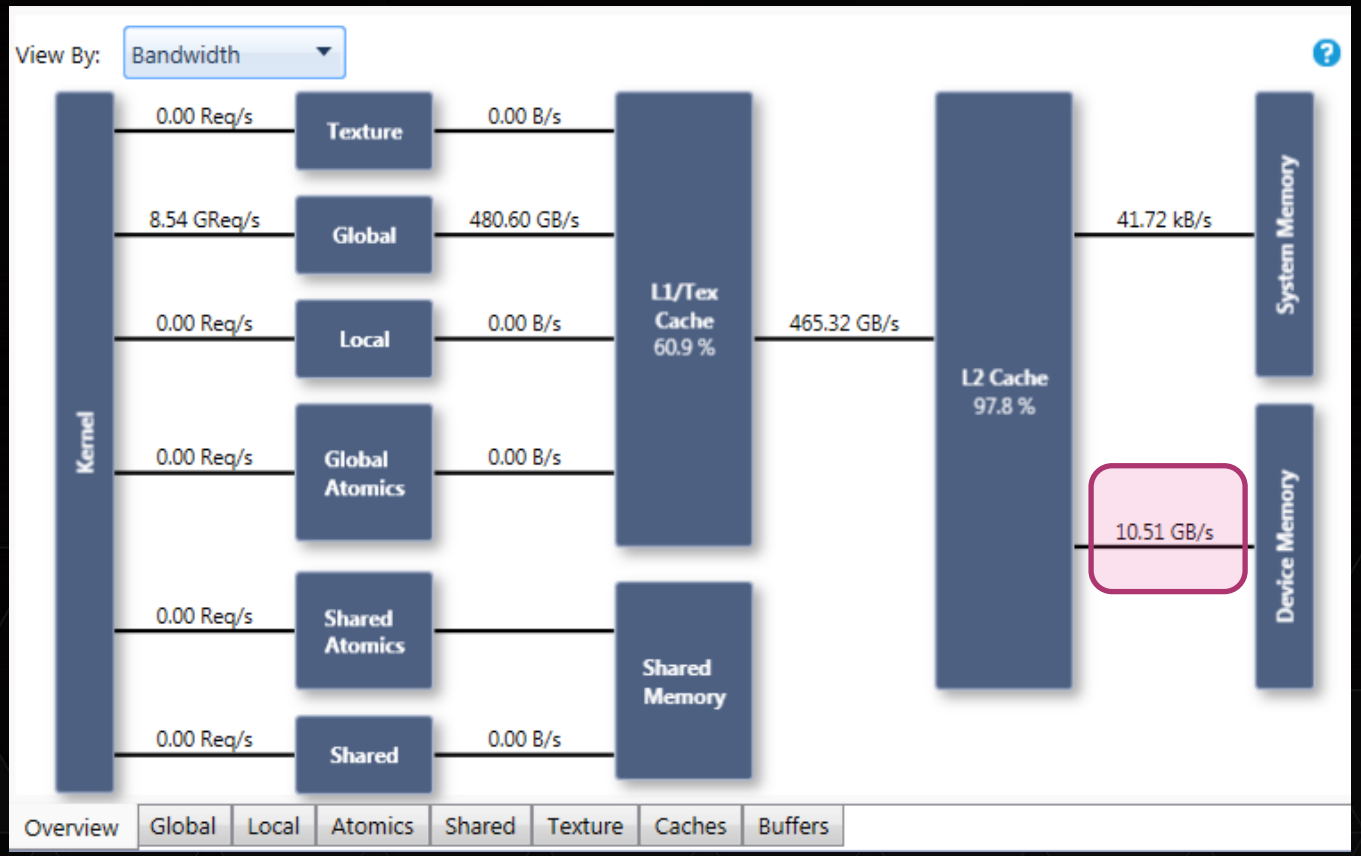
	Function Name	Start Time (µs)	Duration (µs)	Occupancy	Registers per Thread	Grid Dimensions	Block Dimensions	Static Shared Memory per Block (bytes)
1	gaussian_filter_7x7_v0	1,658,458.585	749.120	56.25 %	56	{80, 400, 1}	{32, 4, 1}	0
2	sobel_filter_3x3_v0	2,232,566.488	200.769	100.00 %	17	{80, 200, 1}	{32, 8, 1}	0
3	rgba_to_grayscale_kernel_v0	895,146.264	104.288	100.00 %	8	{80, 200, 1}	{32, 8, 1}	0

- gaussian_filter_7x7_v0() still the hotspot

Kernel	Time	Speedup
Original Version	1.971ms	1.00x
Better Memory Accesses	0.725ms	2.72x

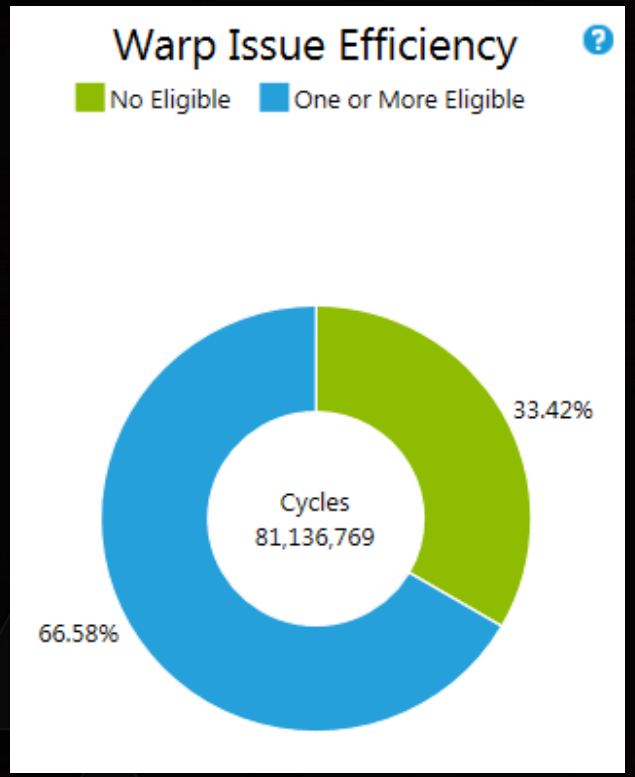
IDENTIFY PERFORMANCE LIMITER

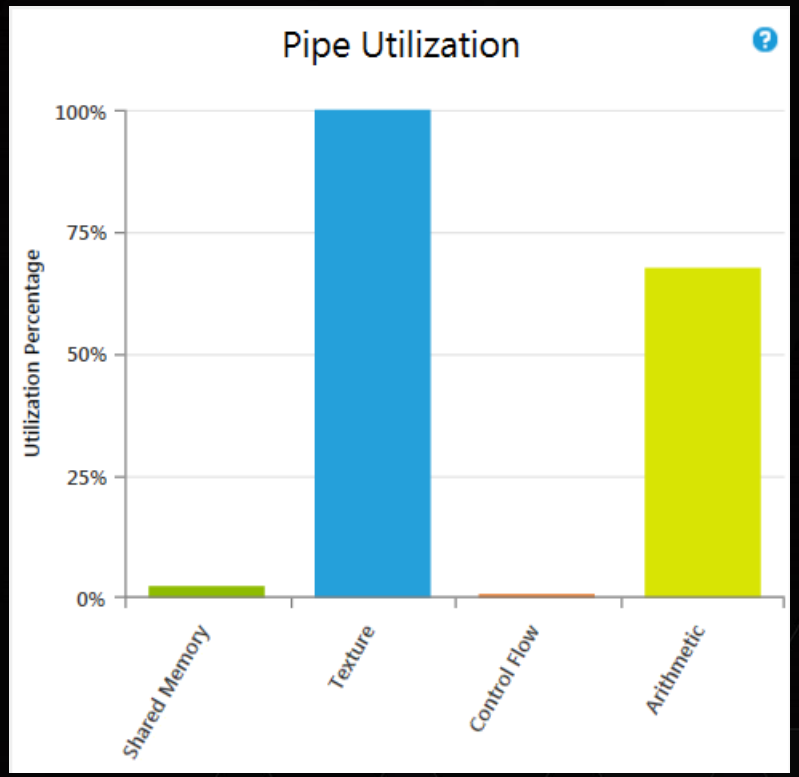
- Utilization of L2$ Bandwidth (BW) limited and DRAM BW < 4%
- Not limited by memory bandwidth



IDENTIFY PERFORMANCE LIMITER

- Scheduler is starting to be busy
- but Tex pipe is clearly the limiter

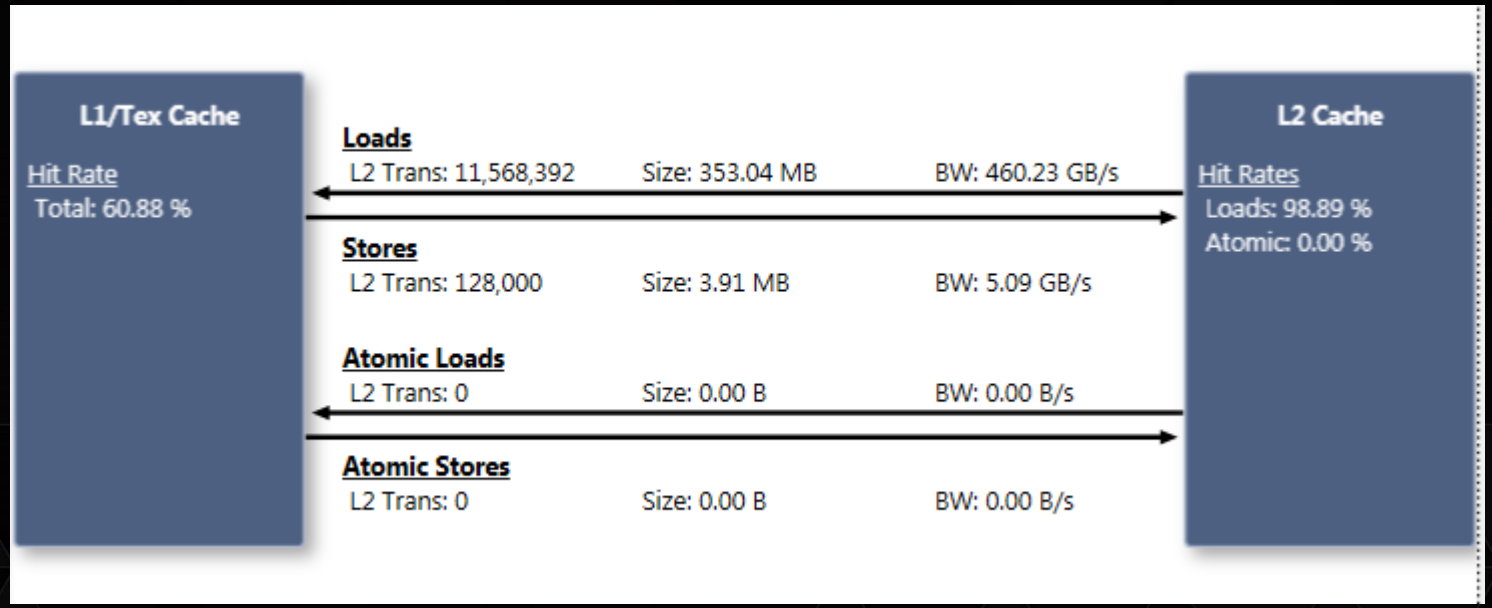




⇒ Load/Store pipeline is saturated

LOOKING FOR INDICATORS

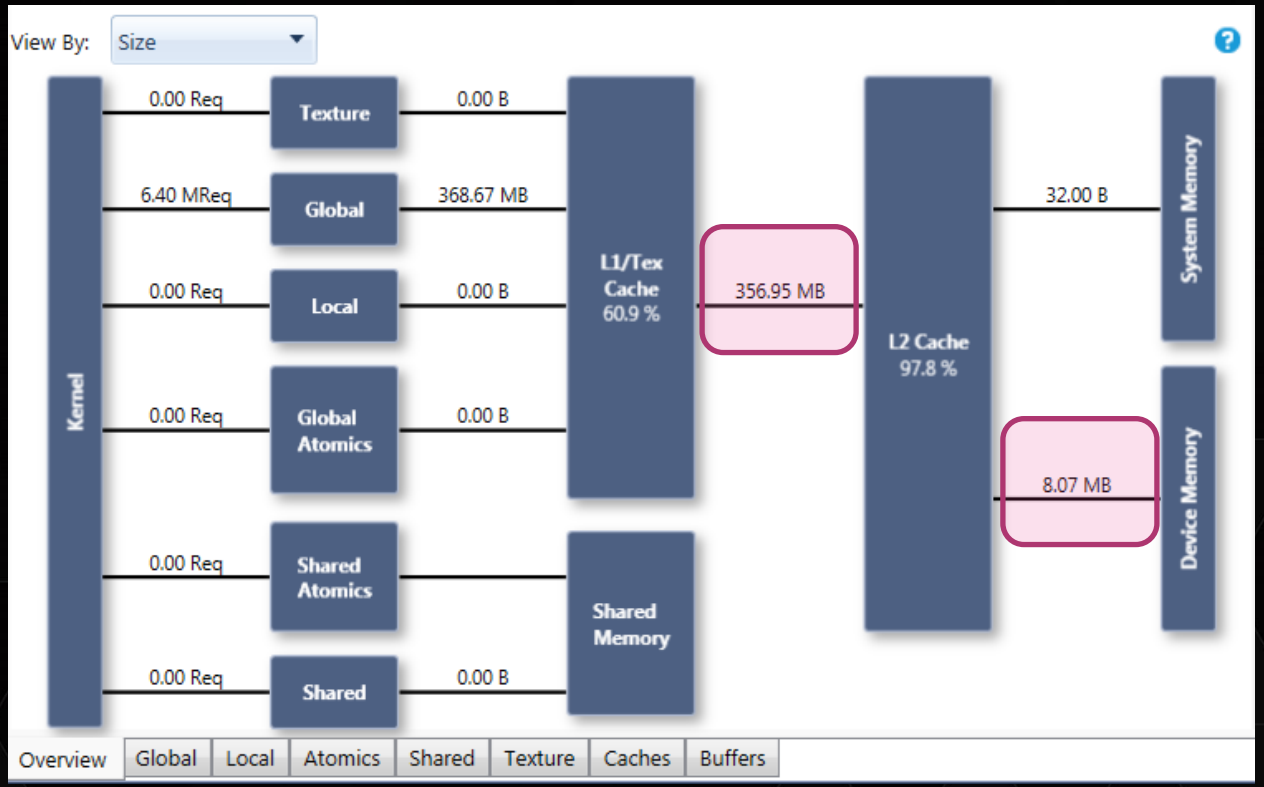
- 98.89% Hit Rate in L2 Cache
- The kernel is mostly working from the L2 cache



GPU TECHNOLOGY CONFERENCE

LOOKING FOR MORE INDICATORS

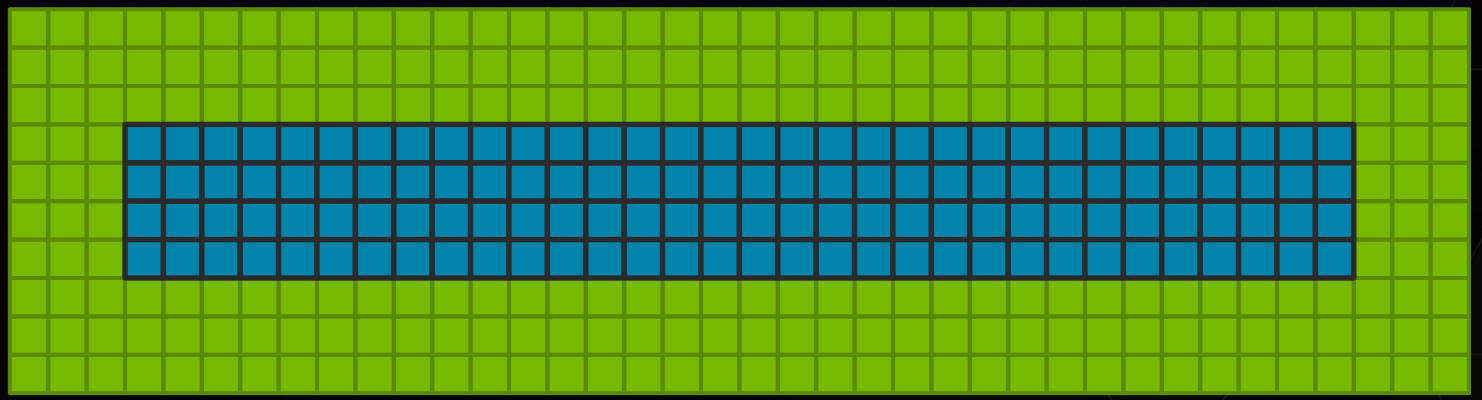
- Kernel Transfers 8MB to/from Device Memory but 360MB to/from L2 Cache



Can we move the data closer to the SM?

SHARED MEMORY

- Adjacent pixels access similar neighbors in Gaussian Filter



- We should use shared memory to store those common pixels

```
__shared__ unsigned char smem_pixels[10][64];
```

GPU TECHNOLOGY CONFERENCE

SHARED MEMORY

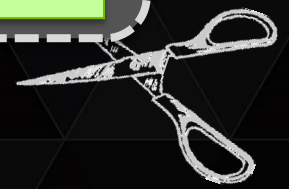
- Using shared memory for the Gaussian Filter
- Significant speedup, < 0.5ms

Kernel	Time	Speedup
Original Version	1.971ms	1.00x
Better Memory Accesses	0.725ms	2.72x
Shared Memory	0.334ms	5.90x

GPU TECHNOLOGY CONFERENCE

PERF-OPT QUICK REFERENCE CARD

Category:	Latency Bound - Shared Memory
Problem:	Long memory latencies are harder to hide
Goal:	**Reduce** latency, move data to faster memory
Indicators:	Shared memory not occupancy limiter High L2 hit rate Data reuse between threads and small-ish working set
Strategy:	(Cooperatively) move data to: • Shared Memory • (or Registers) • (or Constant Memory) • (or Texture Cache)



PERF-OPT QUICK REFERENCE CARD

Category:	Memory Bound - Shared Memory
Problem:	Too much data movement
Goal:	Reduce amount of data traffic to/from global mem
Indicators:	Higher than expected memory traffic to/from global memory Low arithmetic intensity of the kernel
Strategy:	(Cooperatively) move data closer to SM: • Shared Memory • (or Registers) • (or Constant Memory) • (or Texture Cache)

ITERATION 3

IDENTIFY HOTSPOT

Hotspot

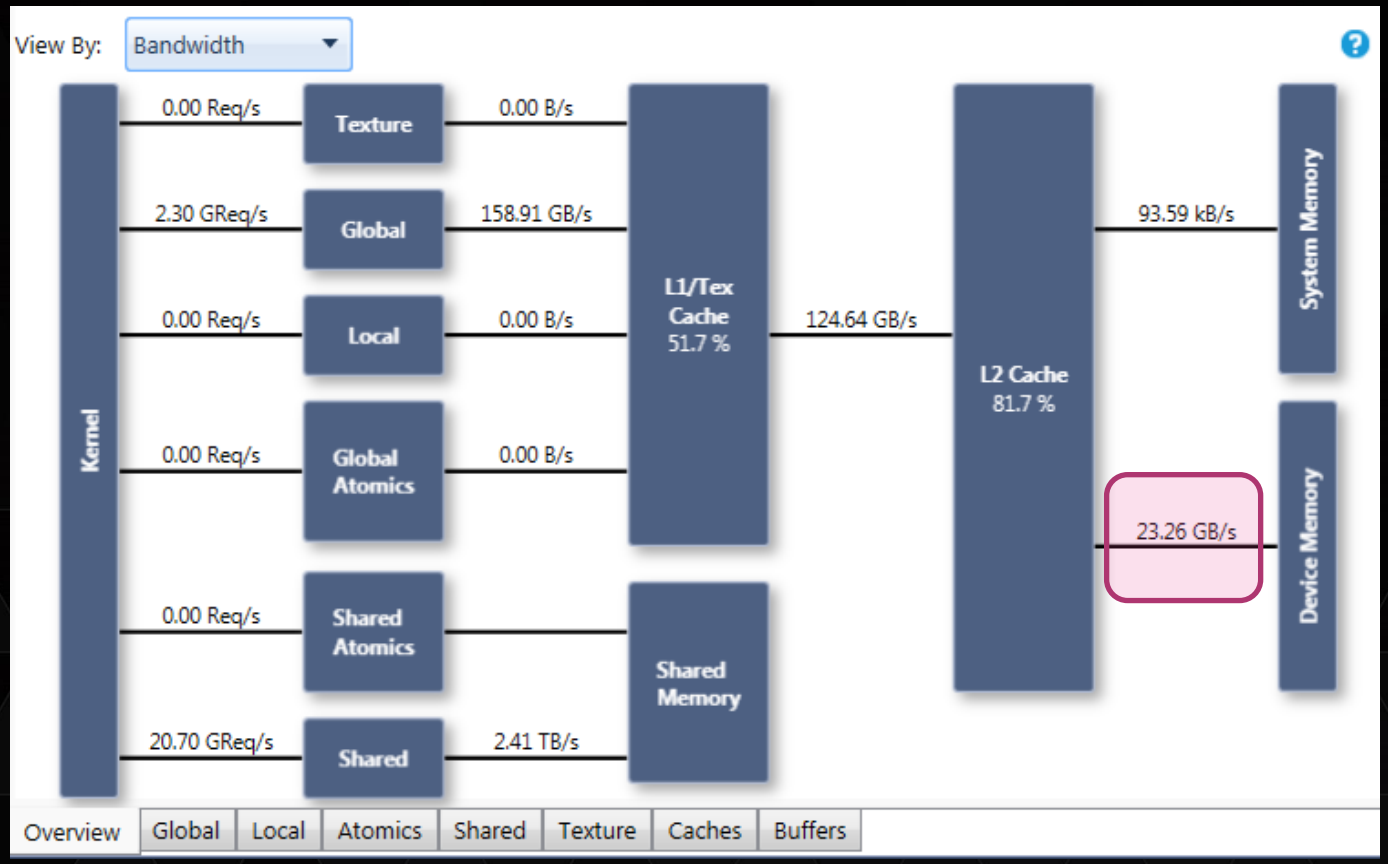
	Function Name	Start Time (µs)	Duration (µs)	Occupancy	Registers per Thread	Grid Dimensions	Block Dimensions	Static Shared Memory per Block (bytes)
1	gaussian_filter_7x7_v2	1,616,699.115	333.920	100.00 %	13	{80, 400, 1}	{32, 4, 1}	640
2	sobel_filter_3x3_v0	2,196,024.459	200.896	100.00 %	17	{80, 200, 1}	{32, 8, 1}	0
3	rgba_to_grayscale_kernel_v0	949,355.531	104.928	100.00 %	8	{80, 200, 1}	{32, 8, 1}	0

- gaussian_filter_7x7_v0() still the hotspot

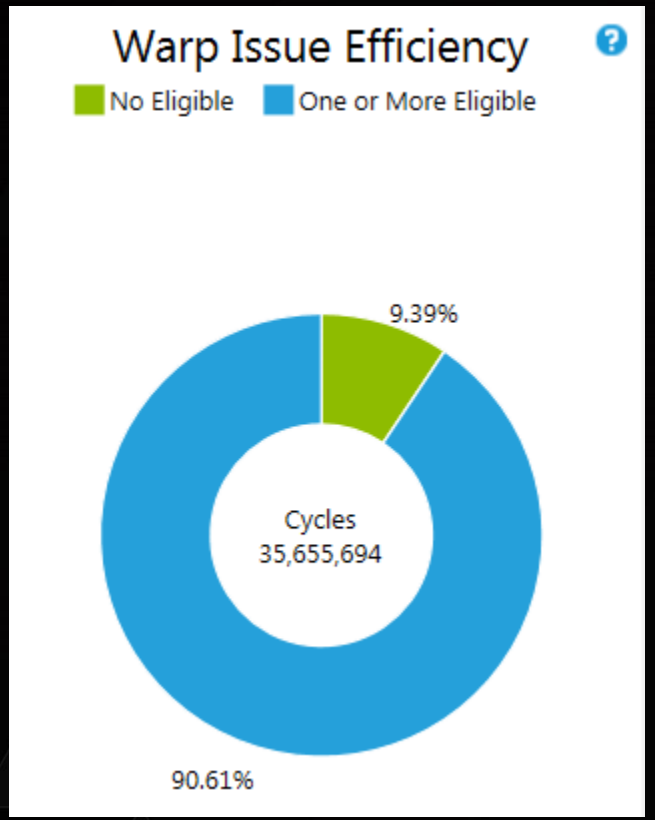
Kernel	Time	Speedup
Original Version	1.971ms	1.00x
Better Memory Accesses	0.725ms	2.72x
Shared Memory	0.334ms	5.90x

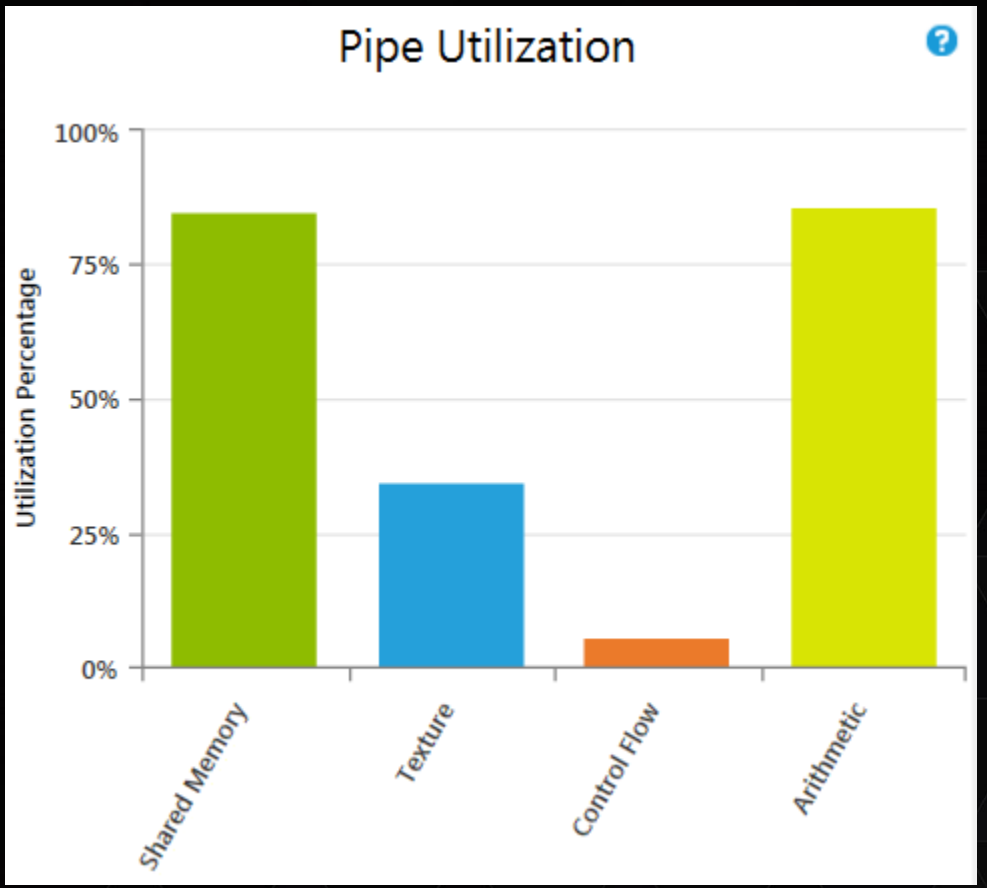
IDENTIFY PERFORMANCE LIMITER

- Utilization of L2$ Bandwidth (BW) moderate and DRAM BW < 8%
- Not limited by memory bandwidth



IDENTIFY PERFORMANCE LIMITER

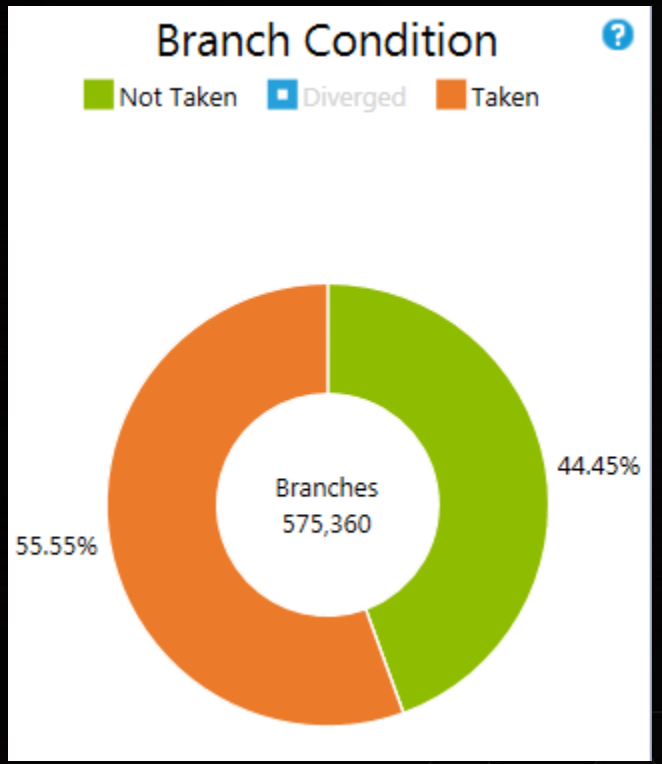




⇒ The Kernel is Compute Bound

LOOKING FOR INDICATORS

- No Divergence in our code



GPU TECHNOLOGY CONFERENCE

BRANCH DIVERGENCE

- Threads of a warp take different branches of a conditional

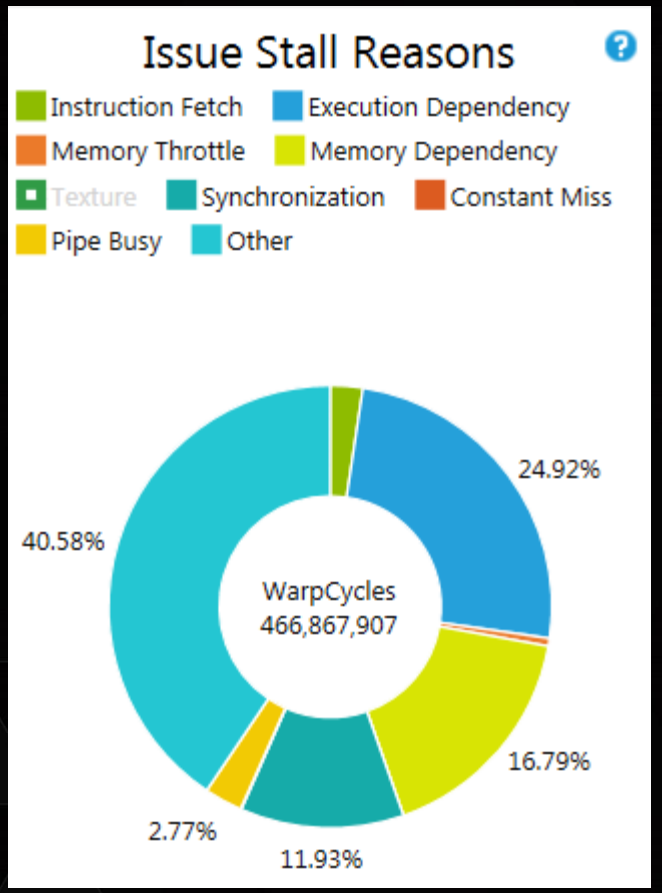
```
if( threadIdx.x < 12 ) {}
```

```
else {}
```

Execution time = “if” branch + “else” branch

Time

LOOKING FOR MORE INDICATORS



- Execution dependency is largest block
- Not a clear indicator however

GPU TECHNOLOGY CONFERENCE

STALL REASONS: EXECUTION DEPENDENCY

- Memory accesses may influence execution dependencies
  - Global accesses create longer dependencies than shared accesses
  - Read-only/texture dependencies are counted in Texture

- Instruction level parallelism can reduce dependencies

```
a = b + c;  // Independent ADDs
d = e + f;
```

GPU TECHNOLOGY CONFERENCE

ILP AND MEMORY ACCESSES

No ILP

```
float a = 0.0f;
for( int i = 0 ; i < N ; ++i )
  a += logf(b[i]);
```

```
c = b[0]
a += logf(c)
c = b[1]
a += logf(c)
c = b[2]
a += logf(c)
c = b[3]
a += logf(c)
```

2-way ILP (with loop unrolling)

```
float a, a0 = 0.0f, a1 = 0.0f;
for( int i = 0 ; i < N ; i += 2 )
{
  a0 += logf(b[i]);
  a1 += logf(b[i+1]);
}
a = a0 + a1
```

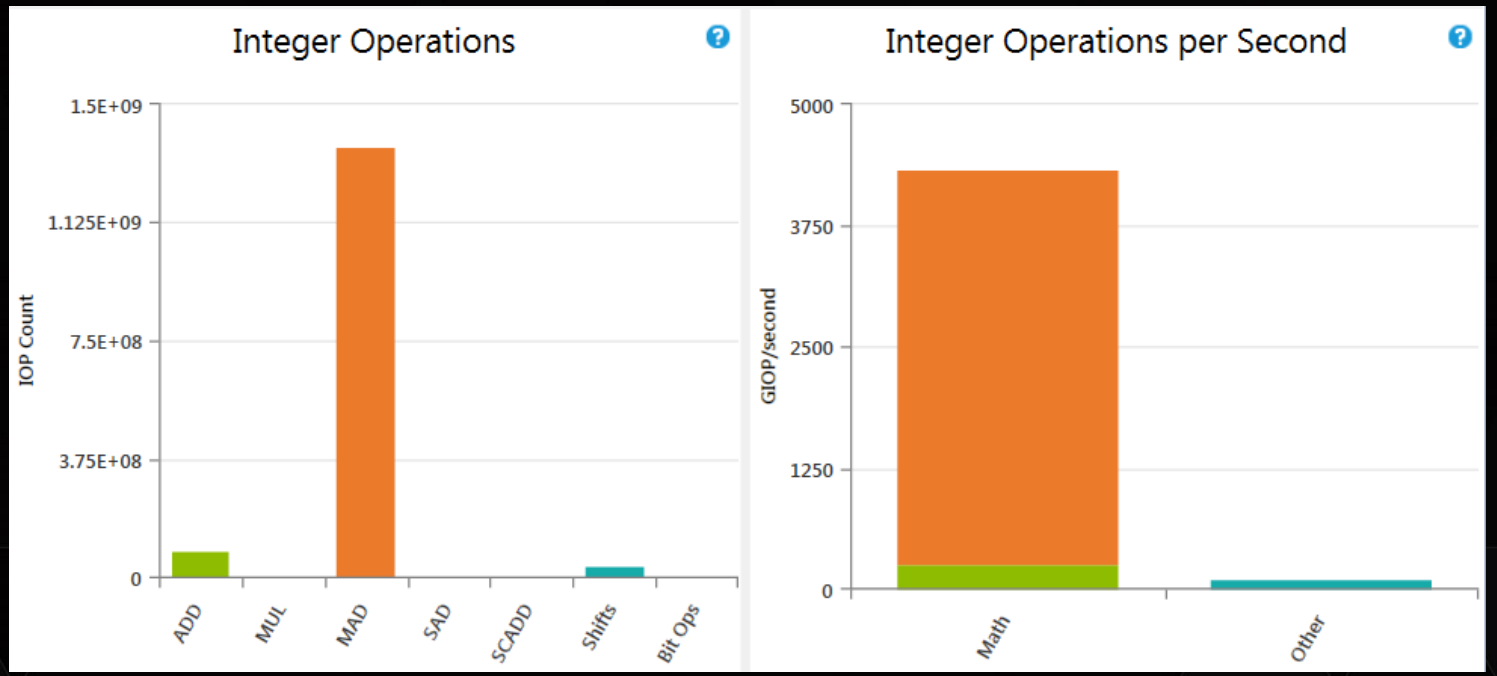
```
c0 = b[0]
a0 += logf(c0)
c0 = b[2]
a0 += logf(c0)
...
a = a0 + a1
```

```
c1 = b[1]
a1 += logf(c1)
c1 = b[3]
a1 += logf(c1)
```

- #pragma unroll is useful to extract ILP
- Manually rewrite code if not a simple loop

LOOKING FOR MORE INDICATORS

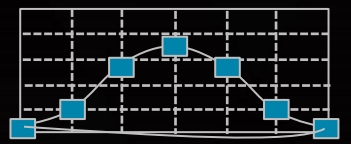
- >4TIOP/second

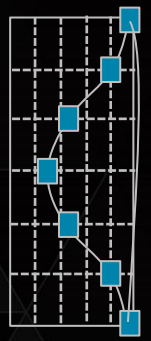


⇒ The Kernel is simply computing a lot

REDUCING COMPUTATIONAL COMPLEXITY

- Separable Filter:
  - Gaussian filters are circular and separable
  - Compute horizontal and vertical convolution separately





1	2	3	4	3	2	1
2	4	6	8	6	4	2
3	6	9	12	9	6	3
4	8	12	16	12	8	4
3	6	9	12	9	6	3
2	4	6	8	6	4	2
1	2	3	4	3	2	1

$$[1\ 2\ 3\ 4\ 3\ 2\ 1] * \begin{bmatrix}1\\2\\3\\4\\3\\2\\1\end{bmatrix} = \begin{bmatrix}1&2&3&4&3&2&1\\2&4&6&8&6&4&2\\3&6&9&12&9&6&3\\4&8&12&16&12&8&4\\3&6&9&12&9&6&3\\2&4&6&8&6&4&2\\1&2&3&4&3&2&1\end{bmatrix}$$

GPU TECHNOLOGY CONFERENCE

SEPARABLE FILTER + INCREASED ILP

- Separable filter reduces computational load
- Processing two elements per thread increases instruction level parallelism

Kernel	Time	Speedup
Original Version	1.971ms	1.00x
Better Memory Accesses	0.725ms	2.72x
Shared Memory	0.334ms	5.90x
Separable Filter + incr. ILP	0.179ms	11.01x

GPU TECHNOLOGY CONFERENCE

PERF-OPT QUICK REFERENCE CARD

Category:	Compute Bound - Branch Divergence
Problem:	Diverging threads
Goal:	Reduce divergence **within** warps
Indicators:	Low warp execution efficiency, high control flow utilization
Strategy:	• Refactor code to avoid intra-warp divergence • Restructure data (sorting?) to avoid data-dependent branch divergence

PERF-OPT QUICK REFERENCE CARD

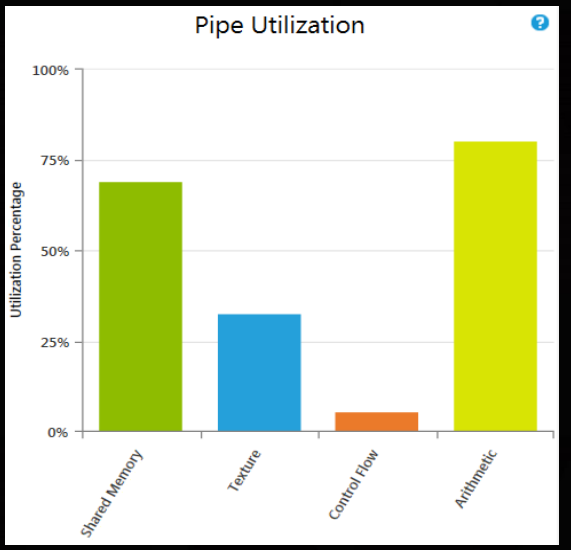
Category:	Latency Bound - Instruction Level Parallelism
Problem:	Not enough independent work per thread
Goal:	Do more parallel work inside single threads
Indicators:	High execution dependency, increasing occupancy has no/little positive effect, still registers available
Strategy:	• Unroll loops (#pragma unroll) • Refactor threads to compute n output values at the same time (code duplication)

GPU TECHNOLOGY CONFERENCE

PERF-OPT QUICK REFERENCE CARD

Category:	Compute Bound - Algorithmic Changes
Problem:	GPU is computing as fast as possible
Goal:	Reduce computation if possible
Indicators:	Clearly compute bound problem, speedup only with less computation
Strategy:	• Pre-compute or store (intermediate) results • Trade memory for compute time • Use a computationally less expensive algorithm

THE RESULT: 11.01X



- Much better utilization
- The sobel filter is starting to become the bottleneck

	Function Name	Start Time (µs)	Duration (µs)	Occupancy	Registers per Thread	Grid Dimensions	Block Dimensions	Static Shared Memory per Block (bytes)
1	sobel_filter_3x3_v0	2,123,269.262	200.512	100.00 %	17	{80, 200, 1}	{32, 8, 1}	0
2	gaussian_filter_7x7_v5	1,537,822.349	179.361	100.00 %	30	{80, 100, 1}	{32, 8, 1}	5120
3	rgba_to_grayscale_kernel_v0	932,662.510	105.056	100.00 %	8	{80, 200, 1}	{32, 8, 1}	0

MORE IN OUR COMPANION CODE

Kernel	Time	Speedup
Gaussian Original Version	1.971ms	1.00x
Better Memory Accesses	0.725ms	2.72x
Shared Memory	0.334ms	5.90x
Separable Filter + incr. ILP	0.179ms	11.01x
Floats instead of int ops	0.153ms	12.88x
Sobel Filter Baseline	0.200ms	1.00x
Floats+Intrinsics+fast_math	0.152ms	1.32x
Your Next Idea!		

GPU TECHNOLOGY CONFERENCE

Companion Code: https://github.com/chmaruni/nsight-gtc2015

SUMMARY

ITERATIVE OPTIMIZATION WITH NSIGHT VSE

- Trace the Application
- Identify the Hotspot and Profile It
- Identify the Performance Limiter
  - Memory Bandwidth
  - Instruction Throughput
  - Latency
- Look for indicators
- Reflect and Optimize the Code
- Iterate

GPU TECHNOLOGY CONFERENCE

REFERENCES

- Performance Optimization: Programming Guidelines and GPU Architecture Details Behind Them, GTC 2013
  - http://on-demand.gputechconf.com/gtc/2013/video/S3466-Performance-Optimization-Guidelines-GPU-Architecture-Details.mp4
  - http://on-demand.gputechconf.com/gtc/2013/presentations/S3466-Programming-Guidelines-GPU-Architecture.pdf
- CUDA Best Practices Guide
  - http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/
- Parallel Forall devblog
  - http://devblogs.nvidia.com/parallelforall/
- Upcoming GTC 2015 Sessions:
  - S5655 CUDA Application Development Life Cycle with Nsight Eclipse Edition (Hands-on lab), Nikita Shulga, Thursday 2pm
  - S5353+S5376 Memory Bandwidth Bootcamp (and Beyond), Tony Scudiero, Thursday 3:30pm and 5pm

GPU TECHNOLOGY CONFERENCE

NVIDIA REGISTERED DEVELOPER PROGRAMS

- Everything you need to develop with NVIDIA products
- Membership is your first step in establishing a working relationship with NVIDIA Engineering
  - Exclusive access to pre-releases
  - Submit bugs and features requests
  - Stay informed about latest releases and training opportunities
  - Access to exclusive downloads
  - Exclusive activities and special offers
  - Interact with other developers in the NVIDIA Developer Forums

REGISTER FOR FREE AT: developer.nvidia.com

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

THANK YOU

JOIN THE CONVERSATION

#GTC15