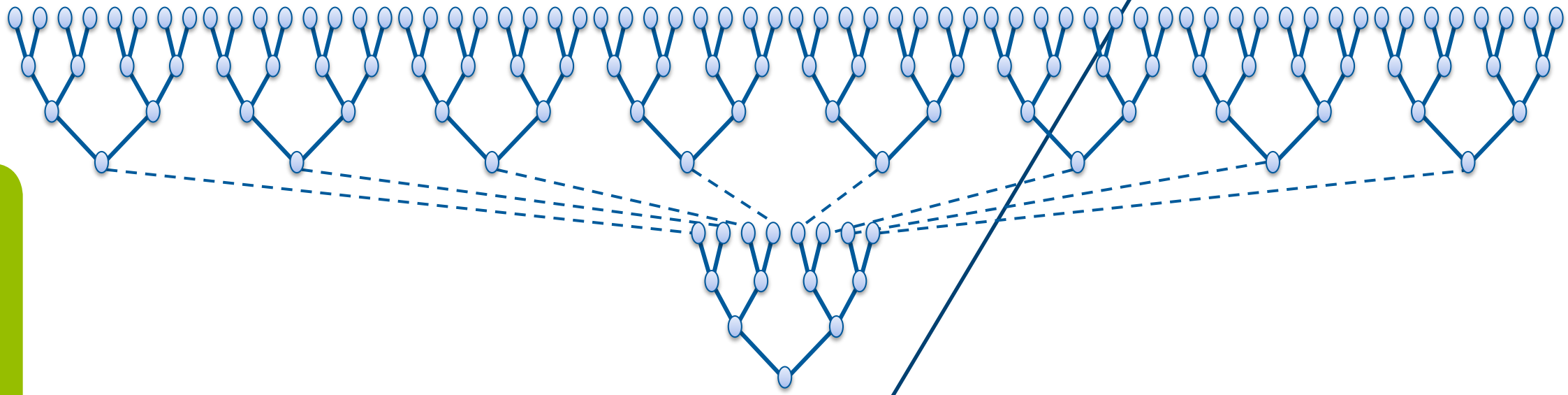
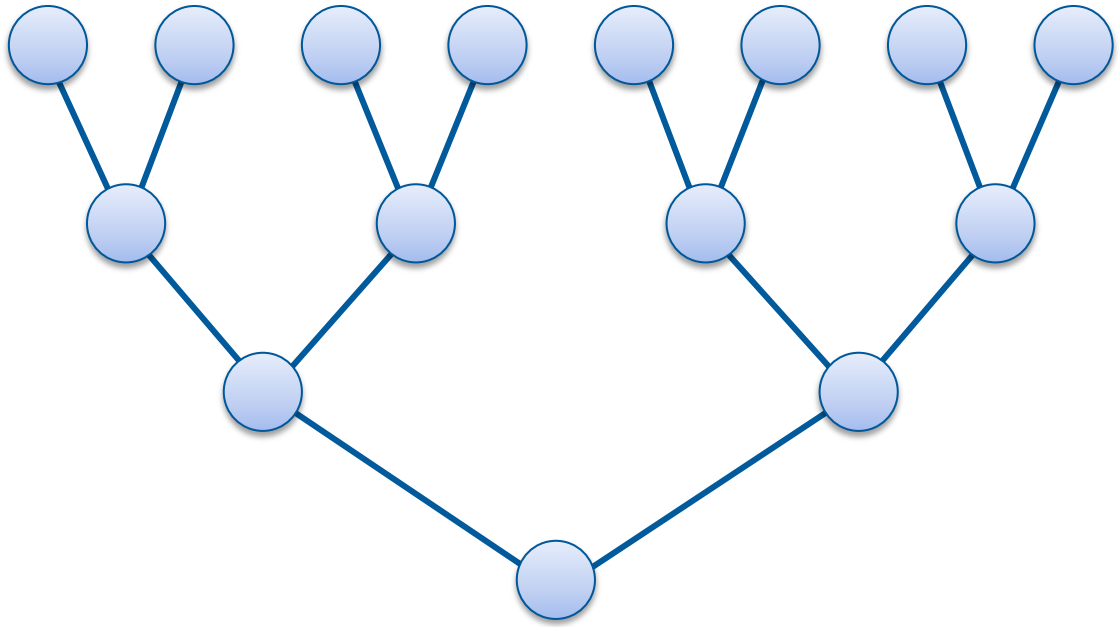


Performance Optimization Case Study: Reductions

10.05.23

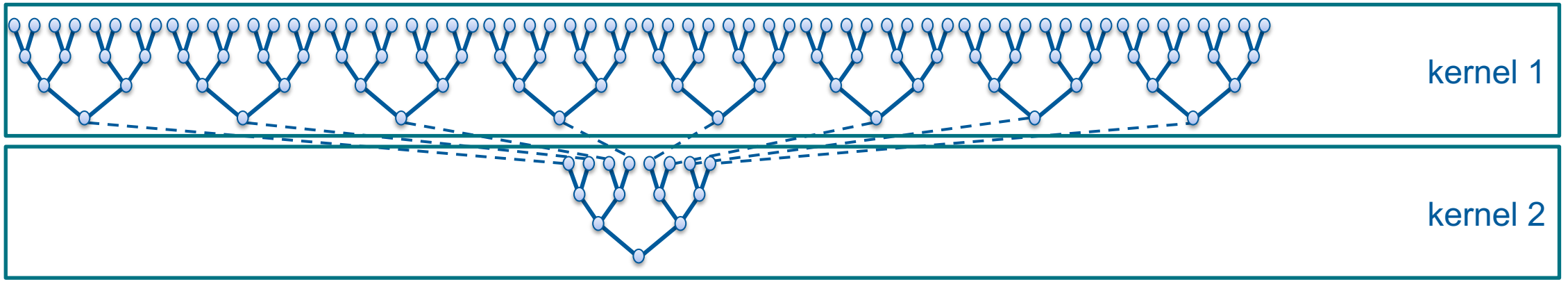
Parallel Reduction

- Common and important data parallel primitive
  - Examples: minimum, maximum, sum
- Many data elements → single output (associative!)
- Easy to implement in CUDA
- Tree-based approach



Parallel Reduction

- Need to use multiple thread blocks
  - To process large arrays
  - To fully utilize GPU
- Partition the array, one block per partition
- How to communicate partial results?



Parallel Reduction #1: Atomic Global

```
__global__ void reduce_atomic_global(const float* input,  float* result, int elements) {
    int id = threadIdx.x + blockIdx.x*blockDim.x;
    atomicAdd(result, input[id]);
}
```

Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	

CPU baseline:
283.989 ms
CPU parallel baseline:
85.393 ms

GTX Titan V peak: 651.3 GB/s

Parallel Reduction #1: Atomic Global

	Instructions	Requests	% Peak	Bank Conflicts
Shared Load	0	0	0	0
Shared Store	0	0	0	0
Shared Atomic	0	-	-	-
Total	0	0	0	0

First-Level Cache

	Instructions	SM->TEX Requests	% Peak	Hit Rate	TEX->L2 Requests	% Peak	L2->TEX Returns	% Peak	TEX->SM Returns	% Peak
Global Load Cached	8,388,608	8,388,608	0.01	0	-	-				
Global Load Uncached	-	-	-	-	-	-	33,554,432	0.05	16,777,216	0.02
Local Load Cached	0	0	0	0	-	-				
Local Load Uncached	-	-	-	-	-	-				
Surface Load	0	0	0	0	-	-	0	0		
Texture Load	0	0	0		-	-	0	0	0	0
Global Store	0	0	0	0			-	-	-	-
Local Store	0	0	0	0	0	0	-	-	-	-
[illegible]	0	0	0	0	0	0				
Global Reduction	8,388,608	8,388,608	0.01	0	268,435,456	0.02	-	-	-	-
[illegible]	0									
Global Atomic	0									
Global Atomic Cas	0	0	0	0	0	0	0	0	see above	see above
Surface Atomic	0									
Surface Atomic Cas	0	0	0	0	0	0	0	0	see above	see above
Loads	8,388,608	8,388,608	0.01	0	-	-	33,554,432	0.05	16,777,216	0.02
Stores	0	0	0	0	0	0	-	-	-	-
Total	16,777,216	16,777,216	0.02	0	268,435,456	0.02	33,554,432	0.05	16,777,216	0.02

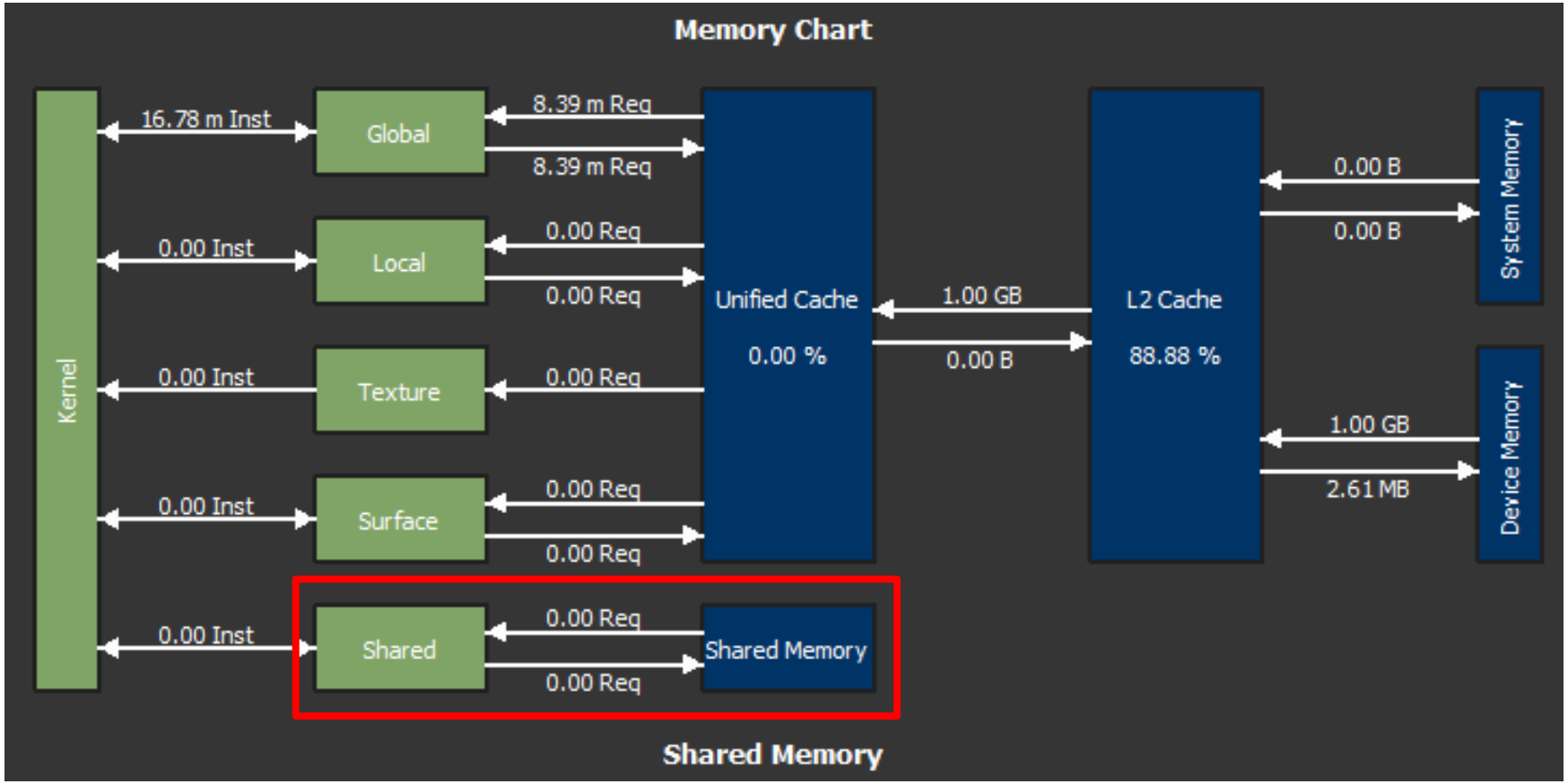
Second-Level Cache

	TEX->L2 Requests	% Peak	L2->TEX Returns	% Peak	Total Bytes	Total Throughput
Global Load Cached	-	-			1,073,741,824	1,519,729,736.71
Global Load Uncached	-	-	33,554,432	0.05	-	-
Local Load Cached	-	-			-	-
Local Load Uncached	-	-			-	-
Surface Load	-	-	0	0	0	0
Texture Load	-	-	0	0	0	0
Global Store	0	0	-	-	0	0
Local Store			-	-		
[illegible]	0	0			0	0
Global Reduction	268,435,456	0.02	-	-	8,589,934,592	12,157,837,893.66
[illegible]	0	0	-	-	0	0
Global Atomic	0	0	0	0	0	0
Global Atomic Cas						
Surface Atomic	0	0	0	0	0	0
Surface Atomic Cas						
Loads	-	-	33,554,432	0.05	1,073,741,824	1,519,729,736.71
Stores	0	0	-	-	0	0
Total	268,435,456	0.02	33,554,432	0.05	9,663,676,416	13,677,567,630.36

Device Memory

	L2<->FB Sectors	% Peak	Bytes	Throughput
Load	33,572,397	0.23	1,074,316,704	1,520,543,398.07
Store	85,286	0.00	2,729,152	3,862,728.78
Total	33,657,683	0.23	1,077,045,856	1,524,406,126.85

Parallel Reduction #1: Atomic Global



Parallel Reduction #2: Atomic Shared

```
__global__ void reduce_atomic_shared(const float* input,  float* result, int elements) {
    int id = threadIdx.x + blockIdx.x*blockDim.x;
    float in = input[id];

    __shared__ float x;
    x = 0.0f;
    __syncthreads();

    atomicAdd(&x, in);
    __syncthreads();

    if (threadIdx.x == 0)
        atomicAdd(result, x);
}
```

Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	

CPU baseline:
283.989 ms
CPU parallel baseline:
85.393 ms

GTX Titan V peak: 651.3 GB/s

Parallel Reduction #2: Atomic Shared

	Instructions	Requests	% Peak	Bank Conflicts
Shared Load	404,404,197	2,842,279,716	98.40	1,020,076
Shared Store	8,388,608	1,903,184,212	65.89	1,028,529
Shared Atomic	943,634,854	-	-	-
Total	1,356,427,659	4,745,463,928	164.29	2,048,605

First-Level Cache

	Instructions	SM->TEX Requests	% Peak	Hit Rate	TEX->L2 Requests	% Peak	L2->TEX Returns	% Peak	TEX->SM Returns	% Peak
Global Load Cached	8,388,608	8,388,608	0.29	0	-	-				
Global Load Uncached	-	-	-	-	-	-	33,554,432	1.16	1,905,957,978	65.99
Local Load Cached	0	0	0	0	-	-				
Local Load Uncached	-	-	-	-	-	-				
Surface Load	0	0	0	0	-	-	0	0		
Texture Load	0	0	0		-	-	0	0	0	0
Global Store	0	0	0	0			-	-	-	-
Local Store	0	0	0	0	0	0	-	-	-	-
Global Reduction	1,048,576	1,048,576	0.04	0	1,048,576	0.00	-	-	-	-
Surface Reduction	0	0	0	0	0	0	-	-	-	-
Global Atomic	0									
Global Atomic Cas	0	0	0	0	0	0	0	0	see above	see above
Surface Atomic	0									
Surface Atomic Cas	0	0	0	0	0	0	0	0	see above	see above
Loads	8,388,608	8,388,608	0.29	0	-	-	33,554,432	1.16	1,905,957,978	65.99
Stores	0	0	0	0	0	0	-	-	-	-
Total	9,437,184	9,437,184	0.33	0	1,048,576	0.00	33,554,432	1.16	1,905,957,978	65.99

Second-Level Cache

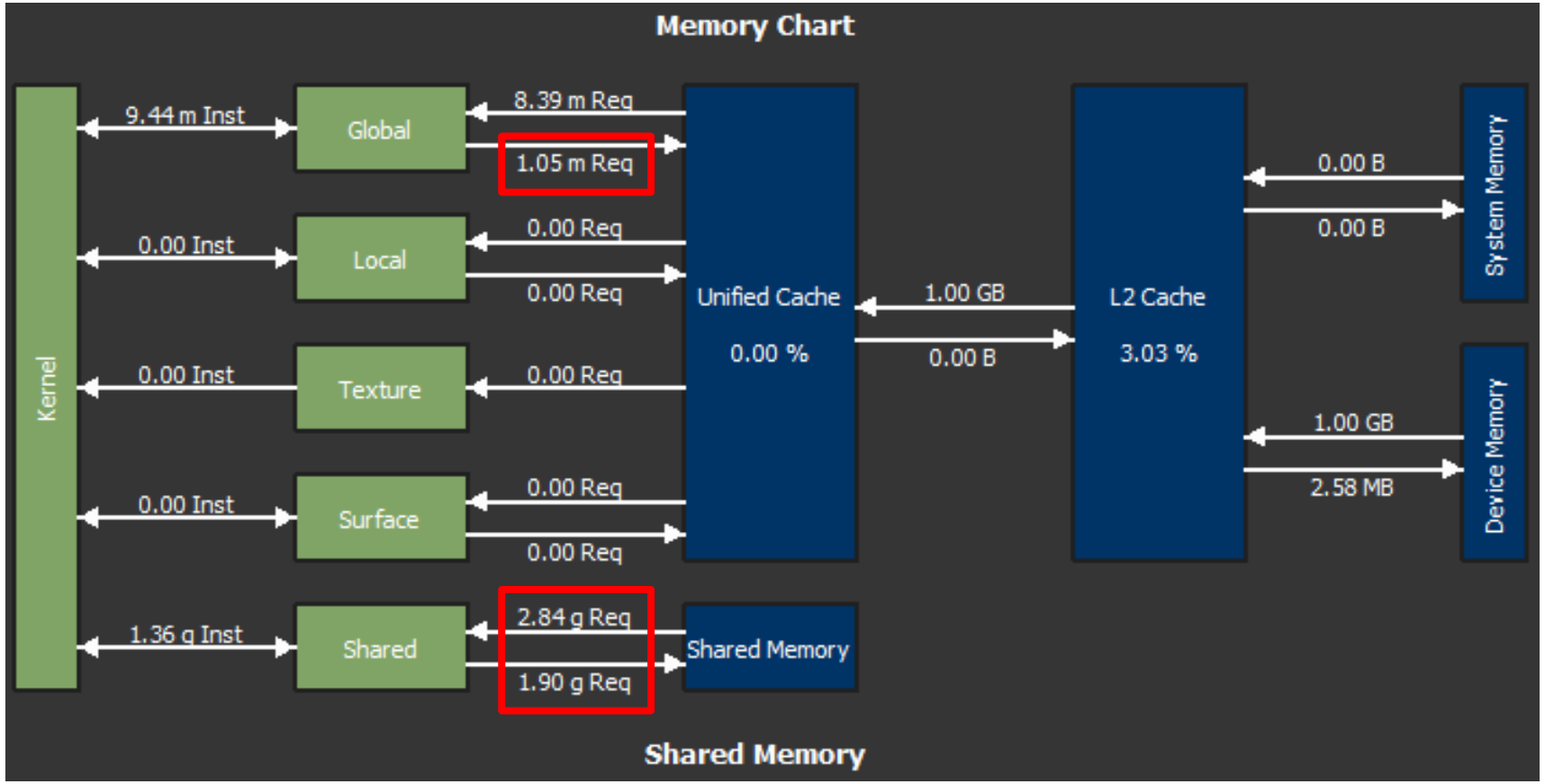
	TEX->L2 Requests	% Peak	L2->TEX Returns	% Peak	Total Bytes	Total Throughput
Global Load Cached	-	-			1,073,741,824	35,673,358,125.97
Global Load Uncached	-	-	33,554,432	1.16	-	-
Local Load Cached	-	-			-	-
Local Load Uncached	-	-			-	-
Surface Load	-	-	0	0	0	
Texture Load	-	-	0	0	0	
Global Store	0	0	-	-		
Local Store			-	-		
Global Reduction	1,048,576	0.00	-	-	33,554,432	1,114,792,441.44
Surface Reduction	0	0	-	-	0	0
Global Atomic	0	0	0	0	0	0
Global Atomic Cas						
Surface Atomic	0	0	0	0	0	0
Surface Atomic Cas						
Loads	-	-	33,554,432	1.16	1,073,741,824	35,673,358,125.97
Stores	0	0	-	-	0	0
Total	1,048,576	0.00	33,554,432	1.16	1,107,296,256	36,788,150,567.40

vs. 8.0 GB

Device Memory

	L2<->FB Sectors	% Peak	Bytes	Throughput
Load	33,555,440	5.46	1,073,774,080	35,674,429,780.08
Store	84,713	0.01	2,710,816	90,062,534.42
Total	33,640,153	5.48	1,076,484,896	35,764,492,314.50

Parallel Reduction #2: Atomic Shared

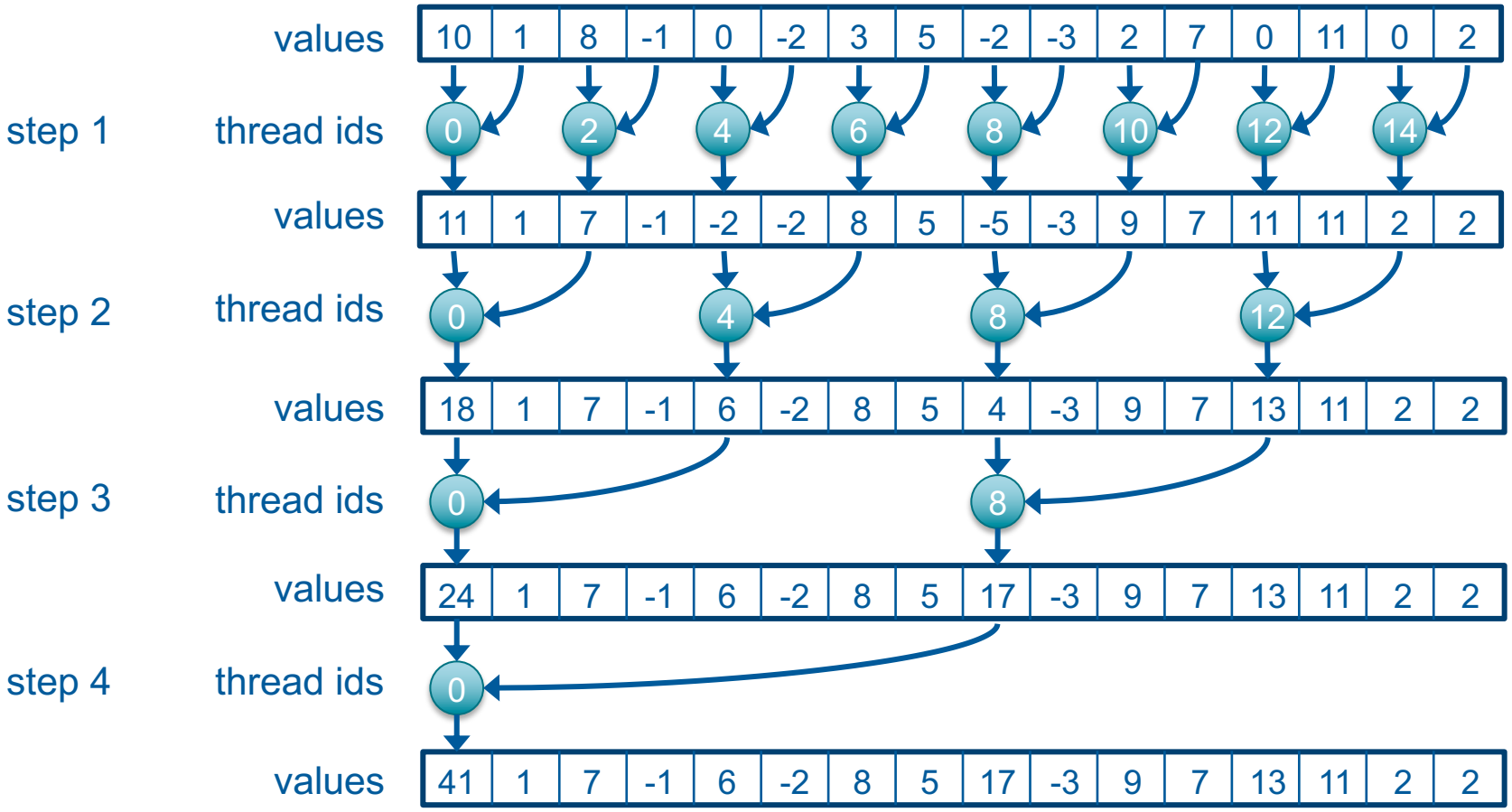


Parallel Reduction #3: Shared

```
__global__ void reduce_shared(const float* input, float* result, int elements) {
    extern __shared__ float data[];
    int id = threadIdx.x + blockIdx.x*blockDim.x;
    data[threadIdx.x] = input[id];
    __syncthreads();

    for (int s = 1; s < blockDim.x; s *= 2) {
        if (threadIdx.x % (2*s) == 0)
            data[threadIdx.x] += data[threadIdx.x + s];
        __syncthreads();
    }
    if (threadIdx.x == 0)
        atomicAdd(result, data[0]);
}
```

Parallel Reduction #3: Shared

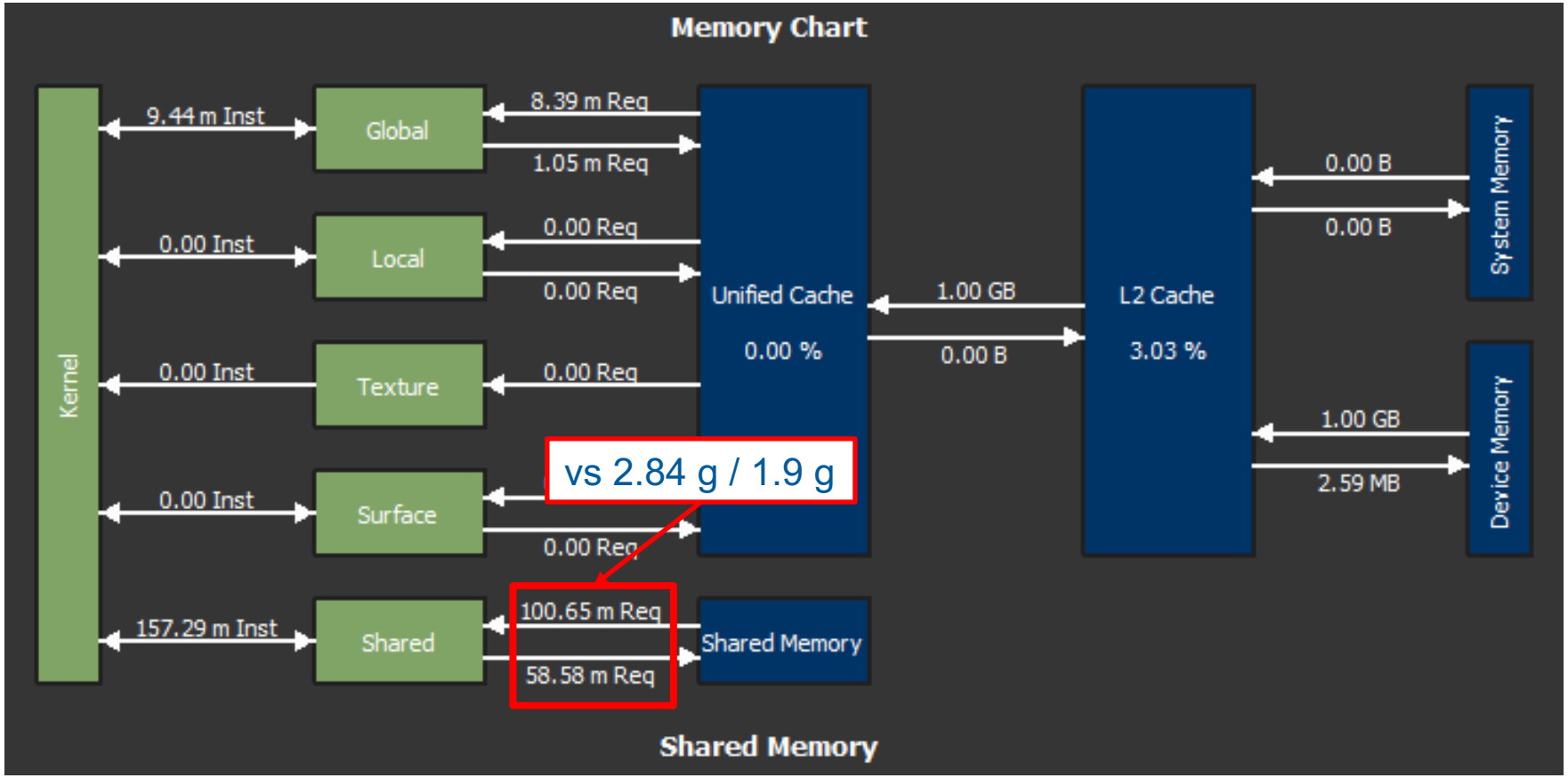


Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	
reduction	6.275 ms	153.726 GB/s	4.288	101.252	1.000

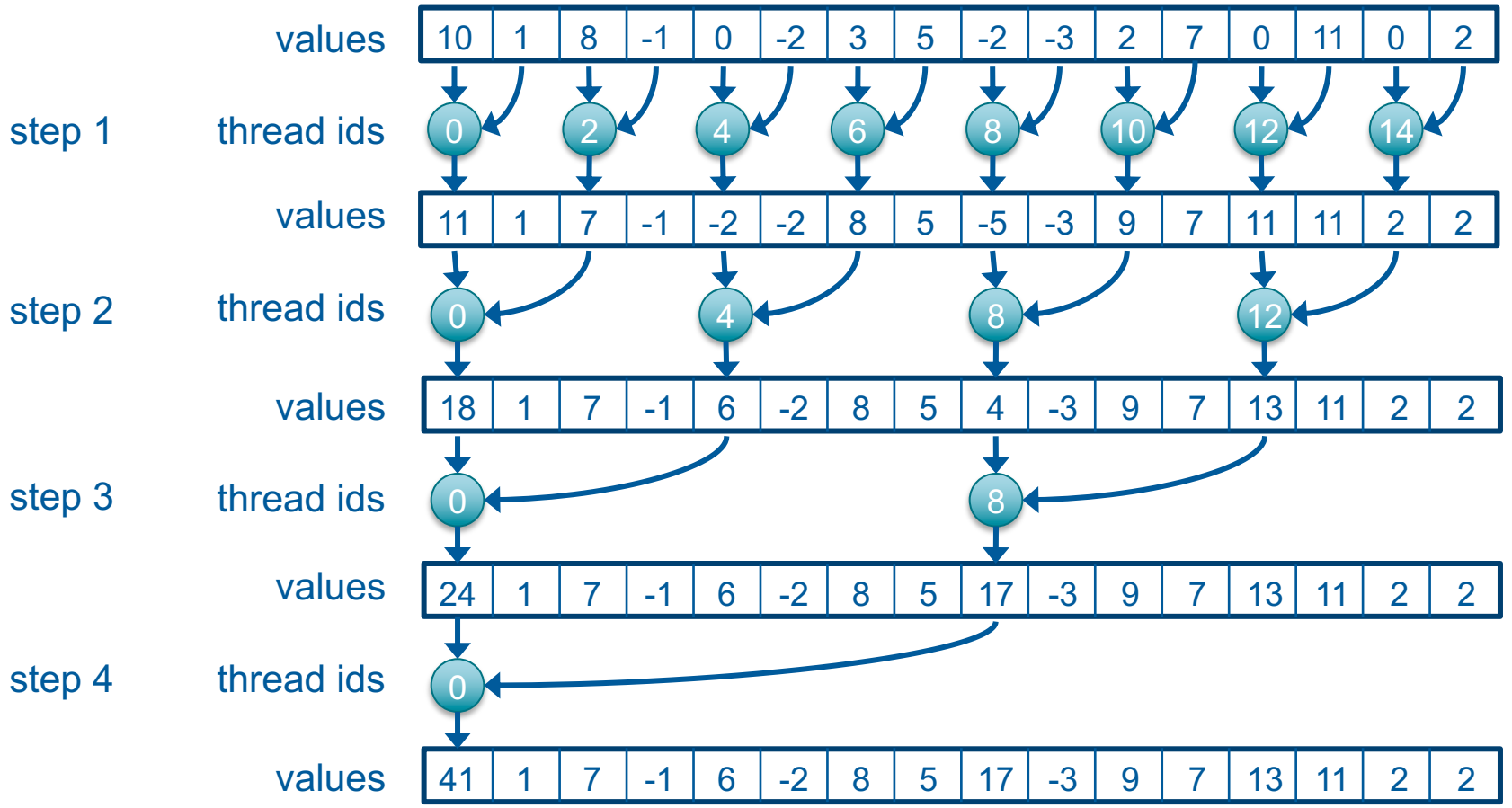
GTX Titan V peak: 651.3 GB/s

Parallel Reduction #3: Shared



Parallel Reduction #3: Shared

- Problem
  - Divergence

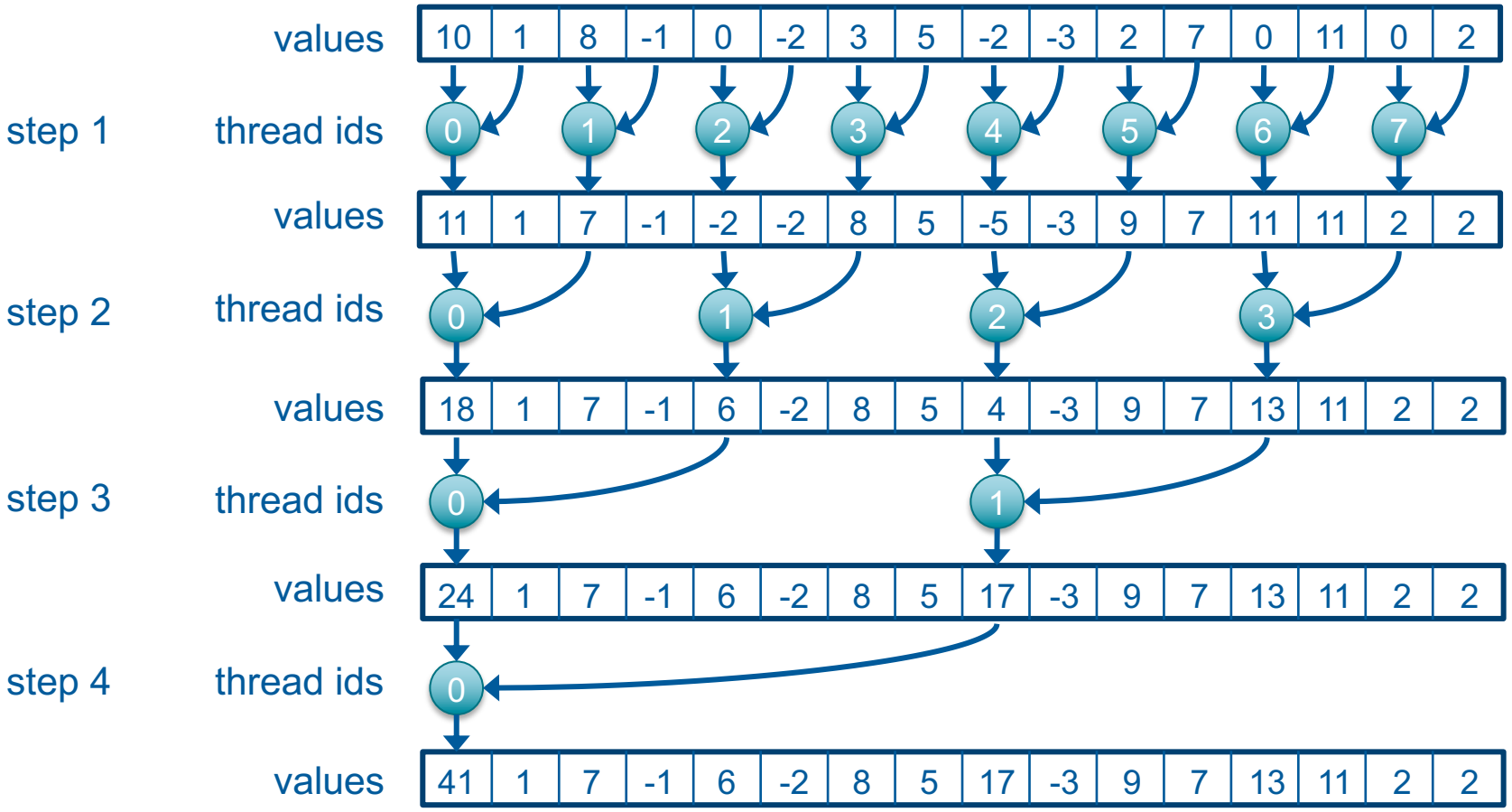


Parallel Reduction #4: Shared Anti Divergence

```
for (int s = 1; s < blockDim.x; s *= 2) {
    if (threadIdx.x % (2*s) == 0)
        data[threadIdx.x] += data[threadIdx.x + s];
    __syncthreads();
}
```

```
for (int s = 1; s < blockDim.x; s *= 2) {
    int lid = 2*s*threadIdx.x;
    if (lid < blockDim.x)
        data[lid] += data[lid + s];
    __syncthreads();
}
```

Parallel Reduction #4: Shared Anti Divergence



Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	
reduction	6.275 ms	153.726 GB/s	4.288	101.252	1.000
anti divergence	3.849 ms	250.859 GB/s	1.630	165.070	1.630

GTX Titan V peak: 651.3 GB/s

Parallel Reduction #4: Shared Anti Divergence

- Problem
  - Bank conflicts

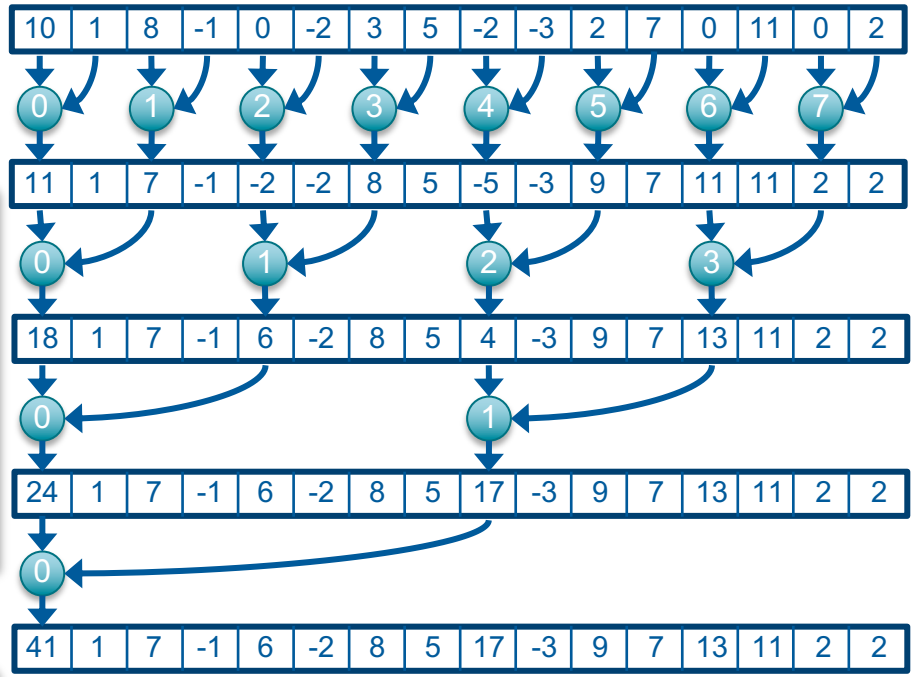
	Instructions	Requests	% Peak	Bank Conflicts
Shared Load	26,214,400	101,061,582	24.55	73,419,745
Shared Store	20,971,520	58,645,681	14.25	37,674,161
Shared Atomic	0	-	-	-
Total	47,185,920	159,707,263	38.80	111,093,906

If multiple addresses of a memory request map to the same memory bank, the accesses are serialized!
Bank conflicts can occur between any threads in the warp!

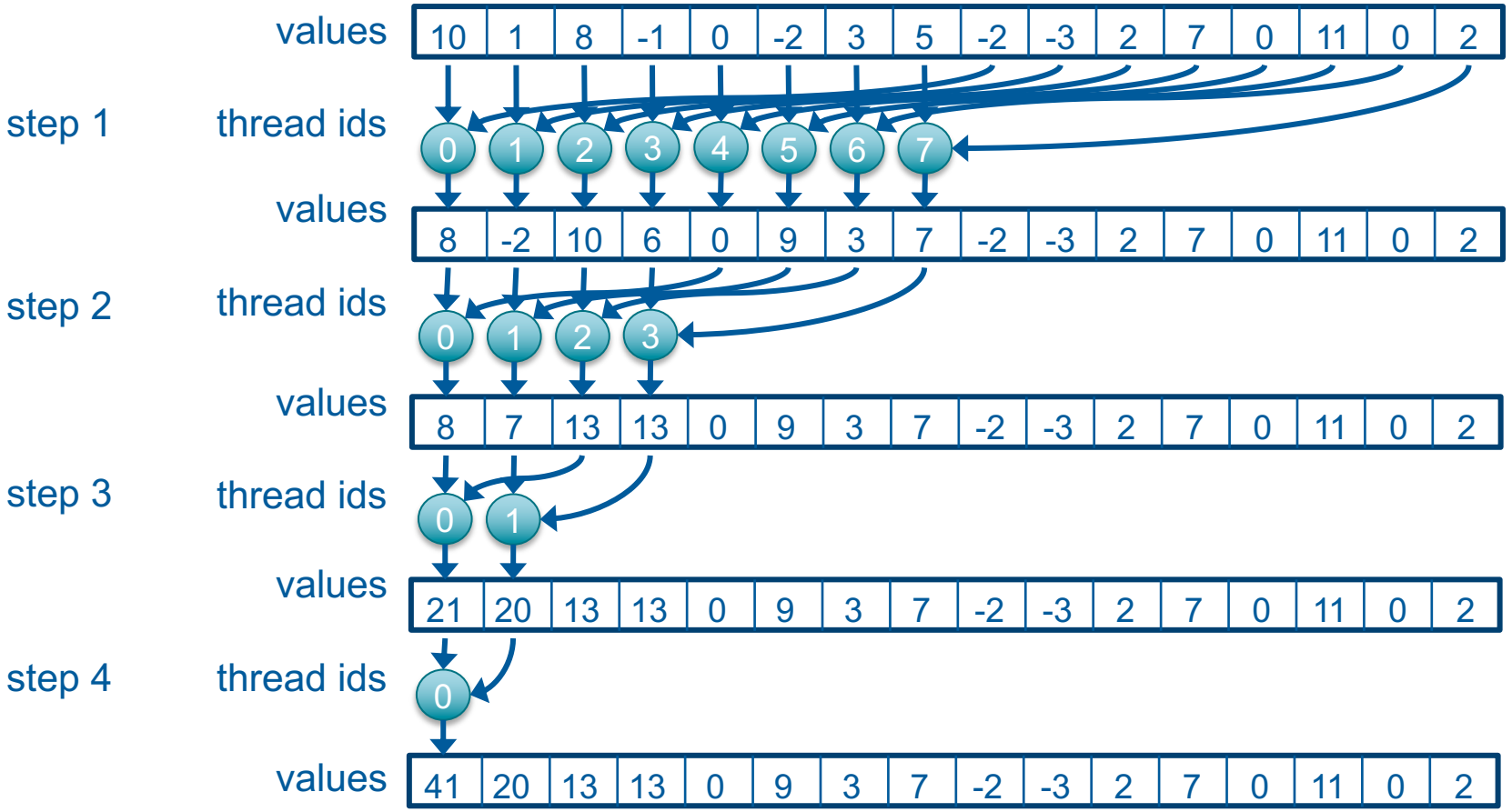
Parallel Reduction #5: Shared Anti Conflicts

```
for (int s = 1; s < blockDim.x; s *= 2) {
    int lid = 2*s*threadIdx.x;
    if (lid < blockDim.x)
        data[lid] += data[lid + s];
    __syncthreads();
}
```

```
for (int s = blockDim.x / 2; s > 0; s /= 2) {
    if (threadIdx.x < s)
        data[threadIdx.x] += data[threadIdx.x + s];
    __syncthreads();
}
```



Parallel Reduction #5: Shared Anti Conflicts



Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	
reduction	6.275 ms	153.726 GB/s	4.288	101.252	1.000
anti divergence	3.849 ms	250.859 GB/s	1.630	165.070	1.630
anti conflicts	2.903 ms	333.536 GB/s	1.326	218.862	2.162

GTX Titan V peak: 651.3 GB/s

Parallel Reduction #5: Shared Anti Conflicts

	Instructions	Requests	% Peak	Bank Conflicts
Shared Load	26,214,400	26,924,195	8.71	63,568
Shared Store	20,971,520	21,875,836	7.08	904,316
Shared Atomic	0	-	-	-
Total	47,185,920	48,800,031	15.79	967,884

vs 111,093,906

Parallel Reduction #5: Shared Anti Conflicts

- Problem
  - Half of threads are idle

```
for (int s = blockDim.x / 2; s > 0; s /= 2) {
    if (threadIdx.x < s)
        data[threadIdx.x] += data[threadIdx.x + s];
    __syncthreads();
}
```

Parallel Reduction #6: Shared Dual Load

```
__global__ void reduce_shared_dual_load(const float* input, float* result, int elements) {
    extern __shared__ float data[];
    int id = threadIdx.x + blockIdx.x*blockDim.x * 2;
    data[threadIdx.x] = input[id] + input[id + blockDim.x];
    __syncthreads();
    for (int s = blockDim.x / 2; s > 0; s /= 2) {
        if (threadIdx.x < s)
            data[threadIdx.x] += data[threadIdx.x + s];
        __syncthreads();
    }
    if (threadIdx.x == 0)
        atomicAdd(result, data[0]);
}
```

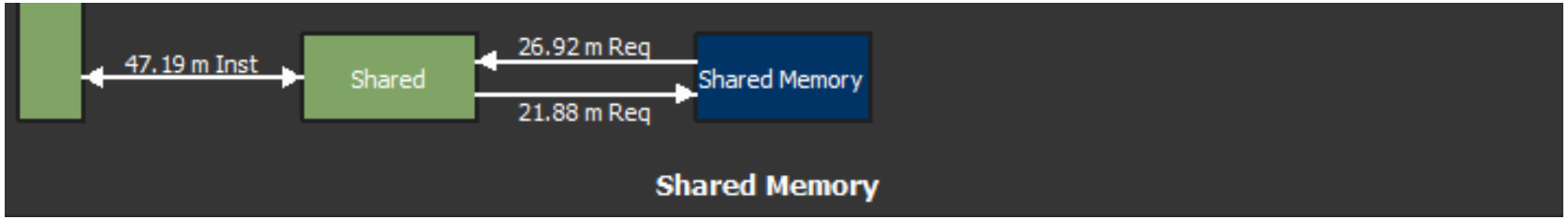
Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	
reduction	6.275 ms	153.726 GB/s	4.288	101.252	1.000
anti divergence	3.849 ms	250.859 GB/s	1.630	165.070	1.630
anti conflicts	2.903 ms	333.536 GB/s	1.326	218.862	2.162
dual load	1.66 ms	612.692 GB/s	1.749	382.744	3.780

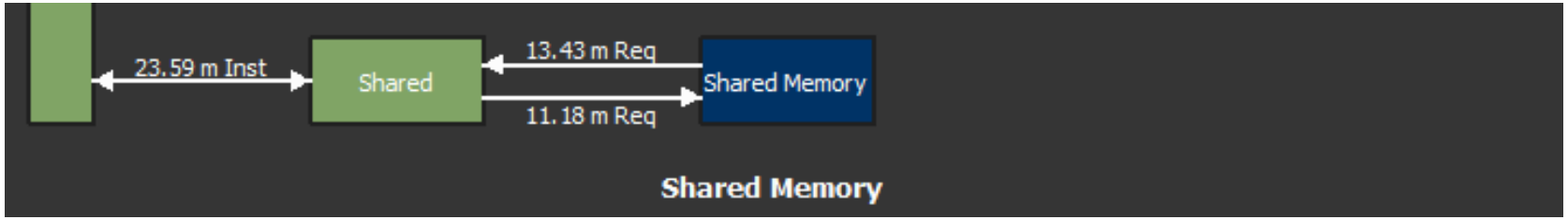
GTX Titan V peak: 651.3 GB/s

Parallel Reduction #6: Shared Dual Load

- `shared_anti_conflicts<<<1048576, 256>>>(…)`



- `shared_dual_load<<<524288, 256>>>(…)`



Parallel Reduction #6: Shared Dual Load

- 612 GB/s vs 651 GB/s peak
- What is the bottleneck?
  - Probably instruction overhead
  - Operations that are not loads, stores, or computations
  - Address arithmetic, loops, conditionals
- Idea: unroll loops

Parallel Reduction #7: Shared Unrolled Last

```
extern __shared__ volatile float data[];
...
for (int s = blockDim.x / 2; s > 32; s /= 2) {
    if (threadIdx.x < s)
        data[threadIdx.x] += data[threadIdx.x + s];
    __syncthreads();
}

if (threadIdx.x < 32) {
    data[threadIdx.x] += data[threadIdx.x + 32];
    data[threadIdx.x] += data[threadIdx.x + 16];
    data[threadIdx.x] += data[threadIdx.x + 8];
    data[threadIdx.x] += data[threadIdx.x + 4];
    data[threadIdx.x] += data[threadIdx.x + 2];
    data[threadIdx.x] += data[threadIdx.x + 1];
}
if (threadIdx.x == 0)
    atomicAdd(result, data[0]);
```

$CC \geq 7.0$

Watch out:
With ITS this requires __syncwarp() after each step!

Parallel Reduction #7: Shared Unrolled Last

```
extern __shared__ float data[];
float x = 0.0f;
...
for (int s = blockDim.x / 2; s > 32; s /= 2) {
    if (threadIdx.x < s)
        data[threadIdx.x] += data[threadIdx.x + s];
    __syncthreads();
}

if (threadIdx.x < 32) {
    x += data[threadIdx.x + 32]; __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 16]; __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 8];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 4];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 2];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 1];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
}
if (threadIdx.x == 0)
    atomicAdd(result, data[0]);
```

Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	
reduction	6.275 ms	153.726 GB/s	4.288	101.252	1.000
anti divergence	3.849 ms	250.859 GB/s	1.630	165.070	1.630
anti conflicts	2.903 ms	333.536 GB/s	1.326	218.862	2.162
dual load	1.66 ms	612.692 GB/s	1.749	382.744	3.780
unrolled last	1.65 ms	646.565 GB/s	1.006	385.063	3.803

GTX Titan V peak: 651.3 GB/s

Parallel Reduction #8: Shared Unrolled All (1/2)

```
template<int BlockSize>
__global__ void reduce_shared_unrolled_all(const float* input, float* result, int elements) {
    extern __shared__ float data[];
    int id = threadIdx.x + blockIdx.x*blockDim.x * 2;
    data[threadIdx.x] = input[id] + input[id + blockDim.x];
    __syncthreads();

    if (BlockSize >= 512) {
        if (threadIdx.x < 256)
            data[threadIdx.x] += data[threadIdx.x + 256];
        __syncthreads();
    }
    if (BlockSize >= 256) {
        if (threadIdx.x < 128)
            data[threadIdx.x] += data[threadIdx.x + 128];
        __syncthreads();
    }
    ...
```

Parallel Reduction #8: Shared Unrolled All (2/2)

```
    if (BlockSize >= 128) {
        if (threadIdx.x < 64)
            data[threadIdx.x] += data[threadIdx.x + 64];
        __syncthreads();
    }

    if (threadIdx.x < 32) {
        x += data[threadIdx.x + 32]; __syncwarp(); data[threadIdx.x] = x; __syncwarp();
        x += data[threadIdx.x + 16]; __syncwarp(); data[threadIdx.x] = x; __syncwarp();
        x += data[threadIdx.x + 8];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
        x += data[threadIdx.x + 4];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
        x += data[threadIdx.x + 2];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
        x += data[threadIdx.x + 1];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    }

    if (threadIdx.x == 0)
        atomicAdd(result, data[0]);
}
```

Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	
reduction	6.275 ms	153.726 GB/s	4.288	101.252	1.000
anti divergence	3.849 ms	250.859 GB/s	1.630	165.070	1.630
anti conflicts	2.903 ms	333.536 GB/s	1.326	218.862	2.162
dual load	1.66 ms	612.692 GB/s	1.749	382.744	3.780
unrolled last	1.65 ms	646.565 GB/s	1.006	385.063	3.803
unrolled all	1.65 ms	646.761 GB/s	1.000	385.063	3.803

GTX Titan V peak: 651.3 GB/s

Parallel Reduction #9: Shared Shuffle

```
if (threadIdx.x < 32) {
    x += data[threadIdx.x + 32]; __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 16]; __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 8];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 4];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 2];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
    x += data[threadIdx.x + 1];  __syncwarp(); data[threadIdx.x] = x; __syncwarp();
}
```

```
if (threadIdx.x < 32) {
    x  = data[threadIdx.x]  +  data[threadIdx.x + 32];
    x += __shfl_sync(0xFFFFFFFF, x, threadIdx.x + 16);
    x += __shfl_sync(0x0000FFFF, x, threadIdx.x + 8);
    x += __shfl_sync(0x000000FF, x, threadIdx.x + 4);
    x += __shfl_sync(0x0000000F, x, threadIdx.x + 2);
    x += __shfl_sync(0x00000003, x, 1);
}
```

Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	
reduction	6.275 ms	153.726 GB/s	4.288	101.252	1.000
anti divergence	3.849 ms	250.859 GB/s	1.630	165.070	1.630
anti conflicts	2.903 ms	333.536 GB/s	1.326	218.862	2.162
dual load	1.66 ms	612.692 GB/s	1.749	382.744	3.780
unrolled last	1.65 ms	646.565 GB/s	1.006	385.063	3.803
unrolled all	1.65 ms	646.761 GB/s	1.000	385.063	3.803
shuffle	1.65 ms	647.059 GB/s	1.000	385.063	3.803

GTX Titan V peak: 651.3 GB/s

Reduction Complexity

$CC \geq 3.0$

- It is possible to directly exchange registers within a warp

```
__shfl_sync(unsigned mask, float val, int src, int width=WarpSize)
```

	Instructions	Requests	% Peak	Bank Conflicts
Shared Load	7,340,032 (-41.67%)	7,470,212 (-41.86%)	3.60 (-41.56%)	16,636 (-52.02%)
Shared Store	7,340,032 (-26.32%)	7,929,460 (-26.15%)	3.82 (-25.77%)	589,428 (-24.01%)
Shared Atomic	0 (+0.00%)	-	-	-
Total	14,680,064 (-34.88%)	15,399,672 (-34.71%)	7.41 (-34.37%)	606,064 (-25.21%)

Parallel Reduction #10: Shared Multi Seq

```
extern __shared__ float data[];
int id = threadIdx.x + blockIdx.x*blockDim.x * 2;
data[threadIdx.x] = input[id] + input[id + threadIdx.x];
__syncthreads();
```

```
extern __shared__ float data[];
int id = threadIdx.x + blockIdx.x*blockDim.x;
int grid_size = BlockSize*gridDim.x;

float in = 0.0f;
for (; id < elements; id += grid_size)
    in += input[id];

data[threadIdx.x] = in;
__syncthreads();
```

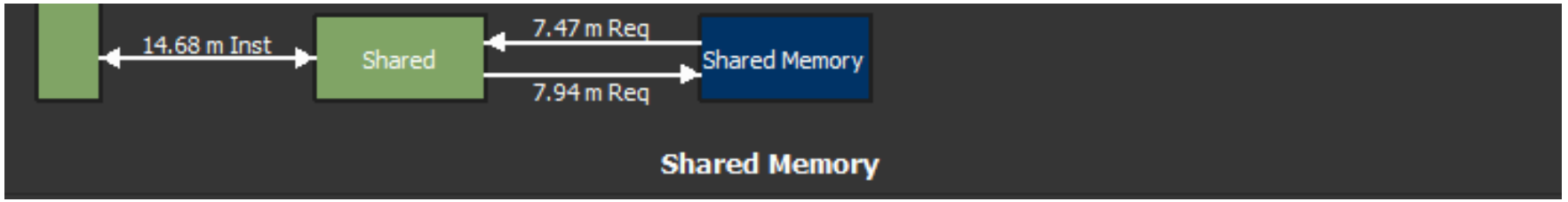
Reduction Performance: 2^{28} = 268 Million Elements

	Time (2^{28} floats)	Bandwidth	Step speedup	Cumulative speedup	Speedup to reduction
atomic global	635.355 ms	1.524 GB/s	1.000	1.000	
atomic shared	26.911 ms	35.764 GB/s	23.609	23.609	
reduction	6.275 ms	153.726 GB/s	4.288	101.252	1.000
anti divergence	3.849 ms	250.859 GB/s	1.630	165.070	1.630
anti conflicts	2.903 ms	333.536 GB/s	1.326	218.862	2.162
dual load	1.66 ms	612.692 GB/s	1.749	382.744	3.780
unrolled last	1.65 ms	646.565 GB/s	1.006	385.063	3.803
unrolled all	1.65 ms	646.761 GB/s	1.000	385.063	3.803
shuffle	1.65 ms	647.059 GB/s	1.000	385.063	3.803
multi seq	1.69 ms	641.734 GB/s	0.976	375.950	3.713

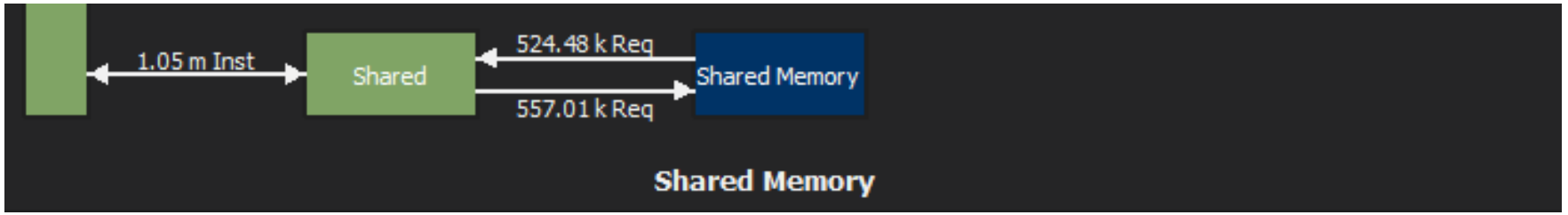
GTX Titan V peak: 651.3 GB/s

Parallel Reduction #10: Shared Multi Seq

- `shared_shuffle<<<524288, 256>>>(…)`



- `shared_multi_seq<<<37450, 256>>>(…)`



Reduction Performance: Different Architectures

	2^{28} GTX 780	2^{28} GTX 970	2^{28} GTX 1080 Ti	2^{28} Titan V	2^{28} RTX 2080 Ti
atomic global	893.79 ms	644.64 ms	451.37 ms	635.35 ms	437.92 ms
atomic shared	558.49 ms	3433.07 ms	649.13 ms	26.91 ms	47.46 ms
reduction	56.72 ms	16.78 ms	17.76 ms	6.28 ms	7.41 ms
anti divergence	42.16 ms	38.50 ms	10.62 ms	3.84 ms	6.02 ms
anti conflicts	31.66 ms	30.45 ms	8.80 ms	2.90 ms	4.83 ms
dual load	17.68 ms	14.35 ms	4.58 ms	1.66 ms	2.53 ms
unrolled last	11.93 ms	10.18 ms	3.08 ms	1.65 ms	1.87 ms
unrolled all	10.37 ms	8.43 ms	3.06 ms	1.65 ms	1.84 ms
shuffle	9.89 ms	8.04 ms	3.03 ms	1.65 ms	1.83 ms
multi seq	7.53 ms	6.95 ms	2.99 ms	1.69 ms	1.87 ms