Efficienet kernel wITH DPC++

ZhuWei, IAGS/MLP/TF GPU

Agenda

- Background
- Efficient Kernel with DPC++
- Q&A

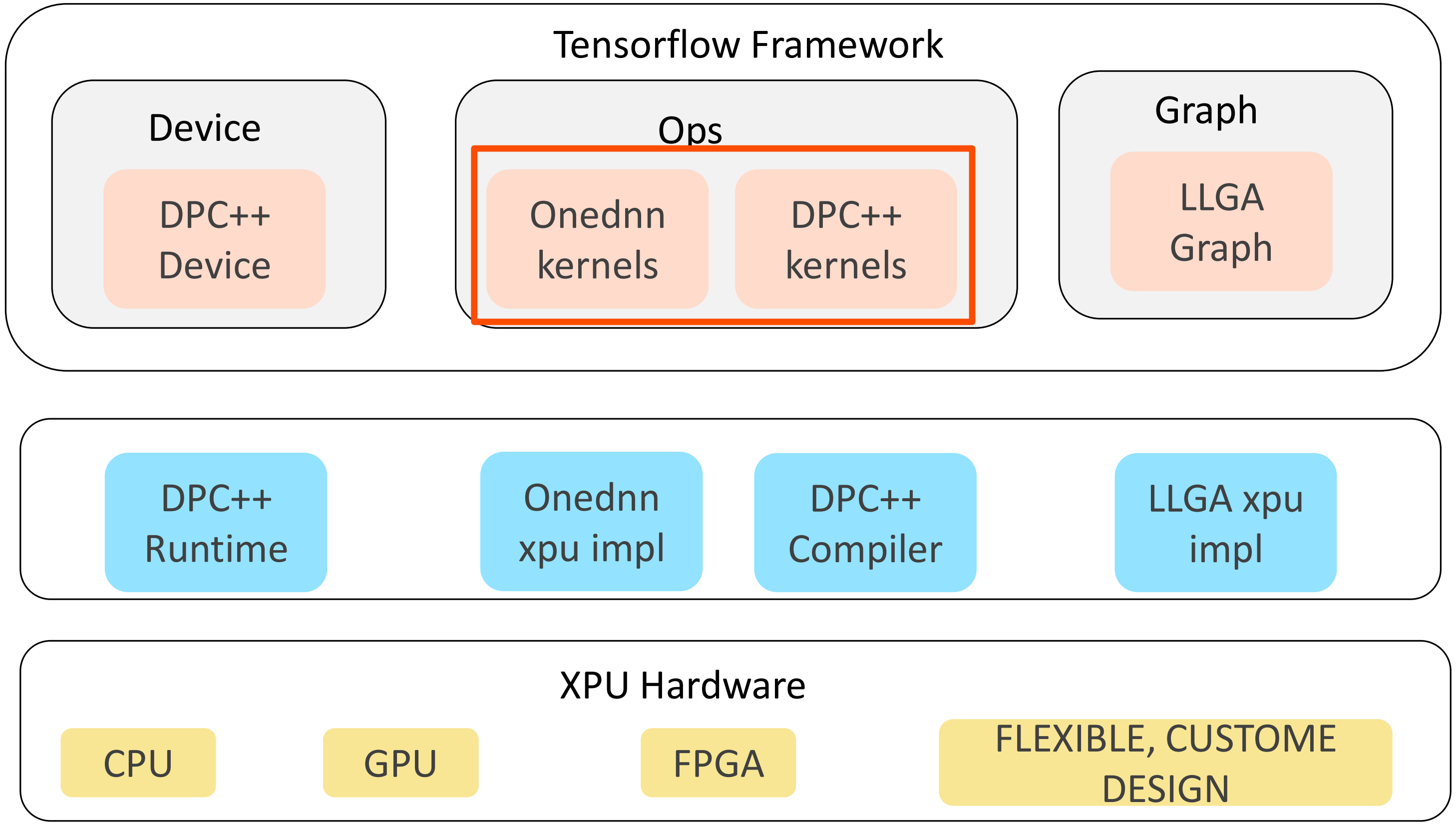
Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Agenda

- Background
- Efficient Kernel with DPC++
- Q&A

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Work Background : TF SoftWare Stack



Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Work Backgroud

- Functionality
  - Op Converage:
    - Enabled 49 ops in dev branch, helped to raise op coverage from 64.39% to 91.62%, met 2020 goal.
    - Enabled 20+ ops in Plugin brach.
- Performance(WIP)
  - Model analysis: rn50, bert
  - Custom Kernel benchmark and Optimization
    - Kernel not supported by Onednn(NMS, L2loss, AdamMomentum and so on)
    - Fusion outside oneDNN

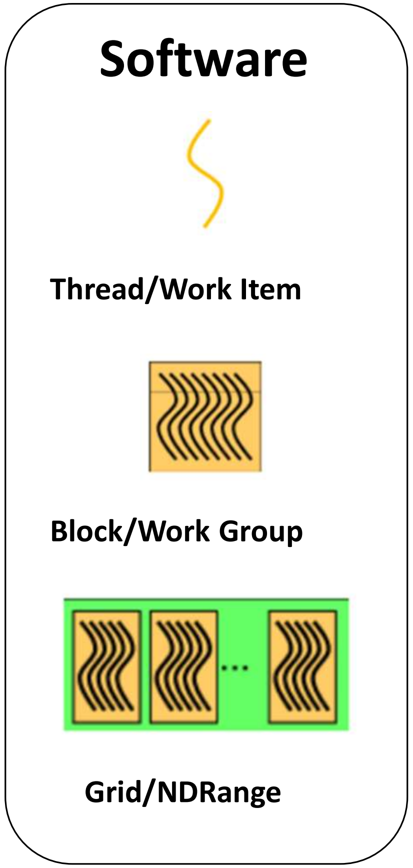
Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

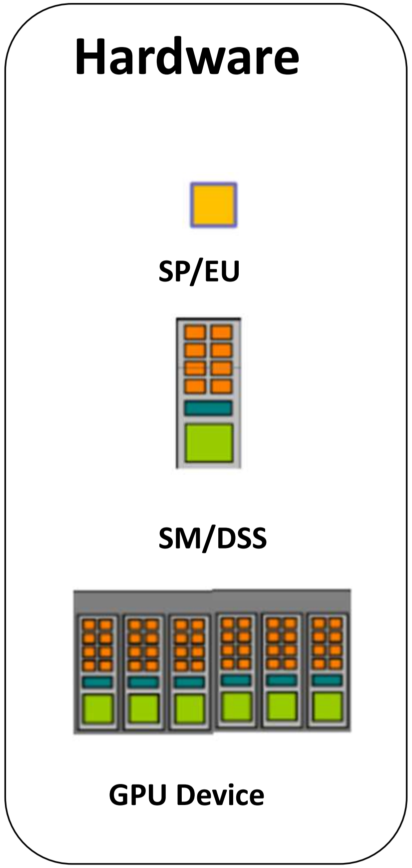
Agenda

- Background
- Efficient Kernel with DPC++
  - Migration from CUDA to DPC++
  - Case study : GEMM optimization
- Q&A

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Migration from CUDA to DPC++: Execution Model

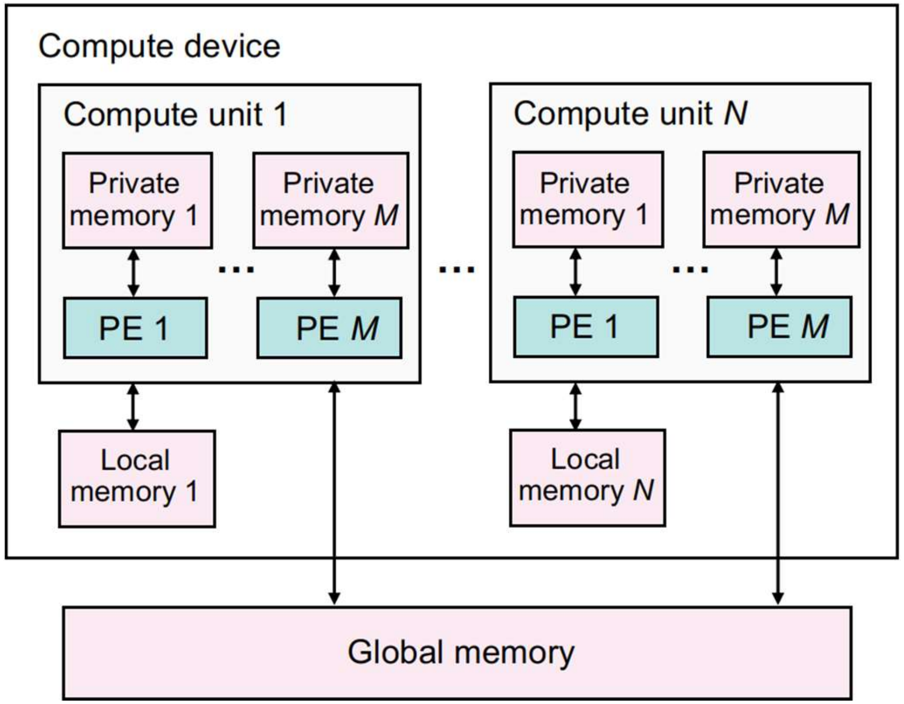




CUDA	DPC++
SP	Process Element
SM	Compute Unit
Thread	Work item
Block	Work group
Grid	NDRange

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Migration from CUDA to DPC++: Memory Model



CUDA	DPC++
Local memory	Private memory
Shared memory	Local memory
Global memory	Global memory
Constant memory	Constant memory

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Efficient Kernel with DPC++

- Case study : GEMM optimization

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

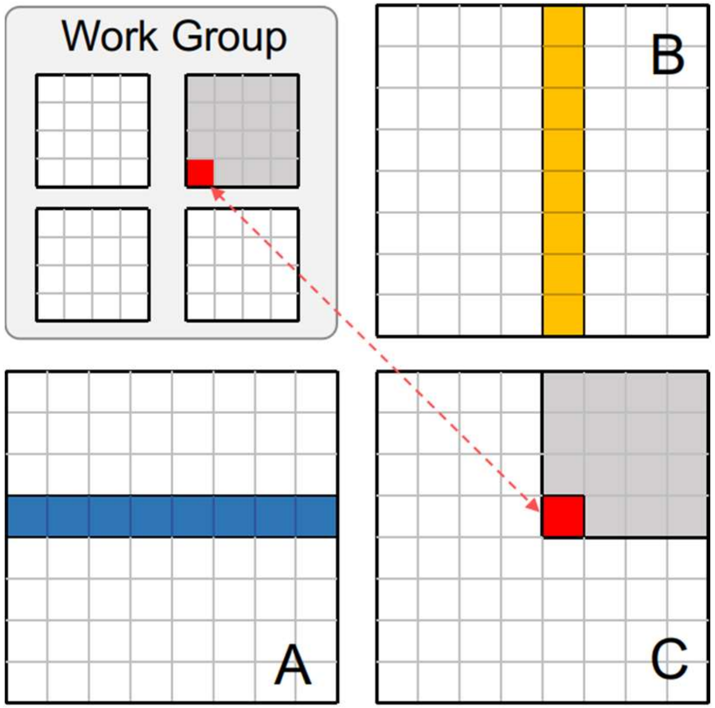
What is Our Optimization Goal

- Strive to reach GPU peak performance
- Choose the right metrics:
  - GFLOP/s: for compute-bound kernels
  - Bandwidth: for memory-bound kernels
- Gemm have high arithmetirc intensity
- Therefore strive for peak GFLOP/s

- Use DG1 for this example
  - 96 EUs, 8 threads/EU, 1100 MHz Clock Frequency
  - 96 * 8 * 1.1 * 2 = 1689 GFLOP/s （Roofline）

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

GEMM #1: Start with DPC++



sycl::malloc_shared

sycl::malloc_device

sycl::aligned_alloc_device

```
buffer buf_a(A.elements, range<2>{M, N});
buffer buf_b(B.elements, range<2>{N, P});
buffer buf_c(C.elements, range<2>{M, P});
queue q;
q.submit([&](handler &cgh) {
  auto a = buf_a.get_access<access::mode::read>(cgh);
  auto b = buf_b.get_access<access::mode::read>(cgh);
  auto c = buf_c.get_access<access::mode::write>(cgh);

  cgh.parallel_for<MatMulKernel>(range<2>{M, P}, [=](id<2> index) {
    size_t row = index[0];
    size_t col = index[1];
    int val = 0.0;
    for (size_t i = 0; i < N; i += 1) {
      val += a[row][i] * b[i][col];
    }
    c[row][col] = val;
  });
});
q.wait();
```

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

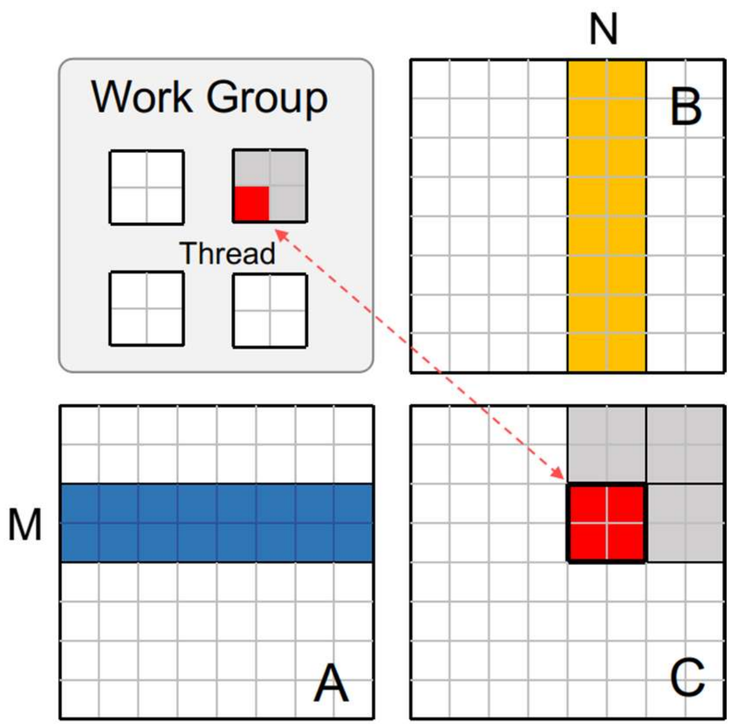
Performance

version	Gflops	Time (ms)	Step Speedup	Cumulative Speedup	Efficiency
v1	141.52	29.63	1.0x	1.0x	8%
Peak	1689	2.49	-	-	

Note: Performance with Matrix size 1280*1280

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

GEMM #2: Multiple Outputs per Thread



Try to increasing cache utilization

```
size_t row = index[0];
size_t col = index[1];
float csub[cm][cn] = {0.0f};
for (int m = 0; m < cm; ++m)
{
  for (int n = 0; n < cn; ++n)
  {
    for (int i = 0; i < N; i += 1)
    {
      csub[m][n] += a[row + m][i] * b[i][col + n];
    }
  }
}
for (int m = 0; m < cm; ++m)
{
  for (int n = 0; n < cn; ++n)
  {
    c[row + m][col + n] += csub[m][n];
  }
}
```

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Performance

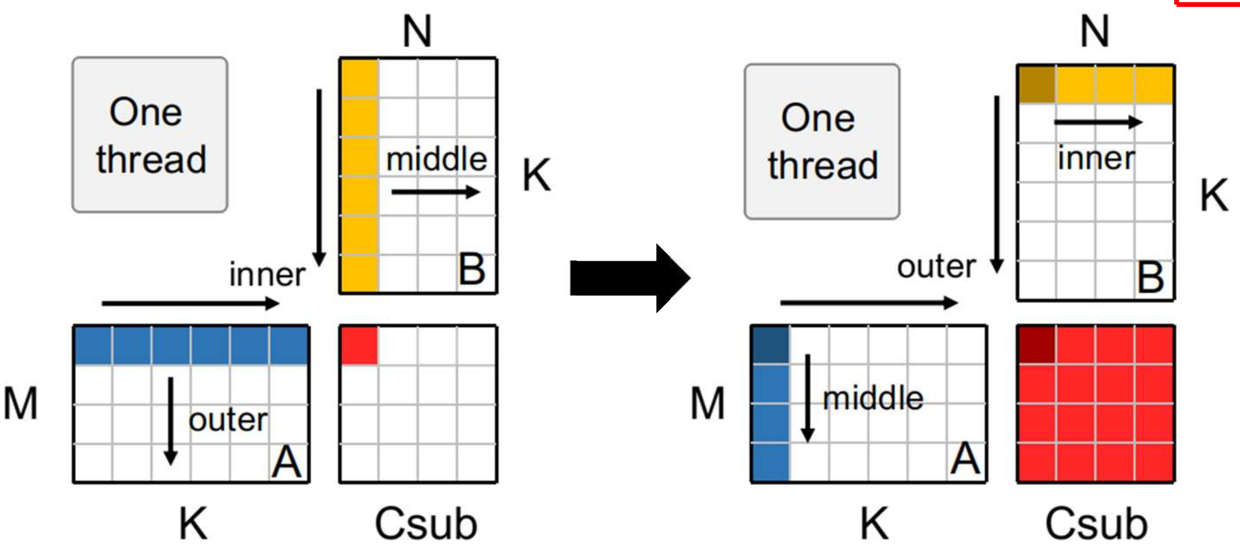
version	Gflops	Time (ms)	Step Speedup	Cumulative Speedup	Efficiency
v1	141.52	29.63	1.0x	1.0x	8%
v2	71.83	58.39	0.51x	0.51x	4.2%

Why performance drop a lot?

Multiple outputs per thread result in less parallelism and latency can't be hided

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

GEMM #3: Multiple Outputs per Thread -- Permute




```
for (int m = 0; m < cm; m++) {
    for (int n = 0; j < cn; n++) {
        for (int i = 0; i < K; i++) {
            c[m][n] += a[row + m][i] * b[i][col + n];
        }
    }
}
```

- Reduce memory repeat access
- Increase L2 cache hit rate

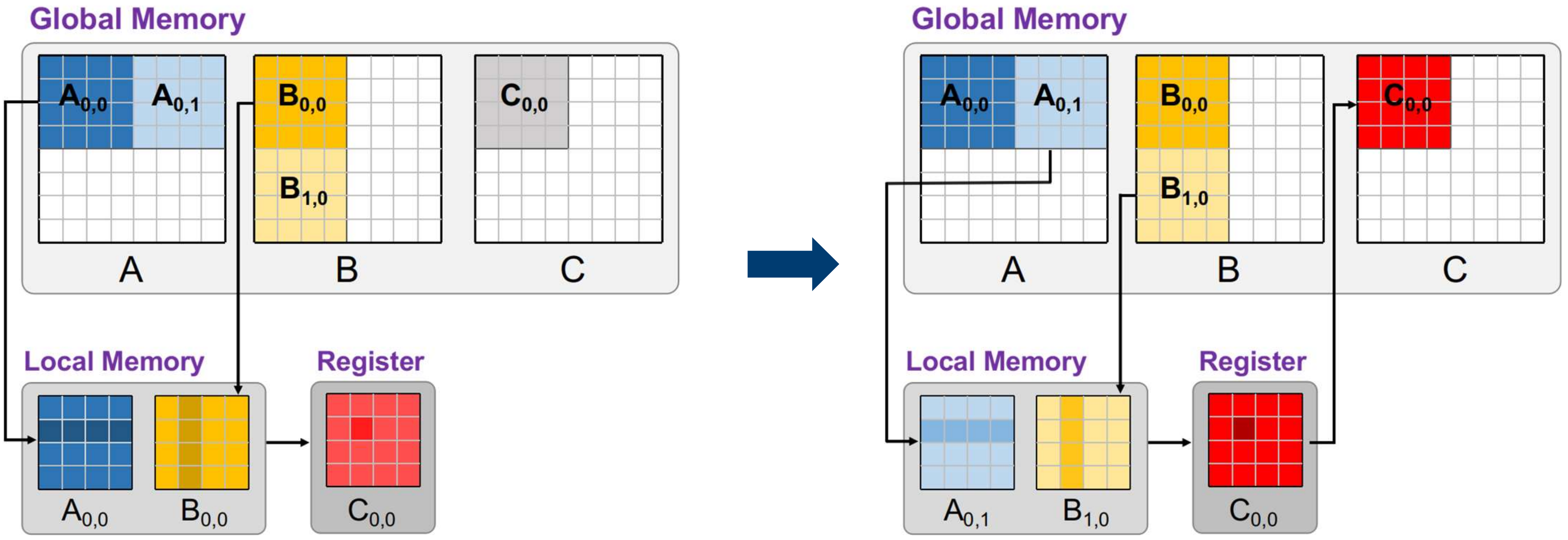
Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Performance

version	Gflops	Time (ms)	Step Speedup	Cumulative Speedup	Efficiency
v1	141.52	29.63	1.0x	1.0x	8.3%
v2	71.83	58.39	0.51x	0.51x	4.2%
v3	609.09	6.88	8.47x	4.31x	36%
Peak	1689	2.49	-	-	

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

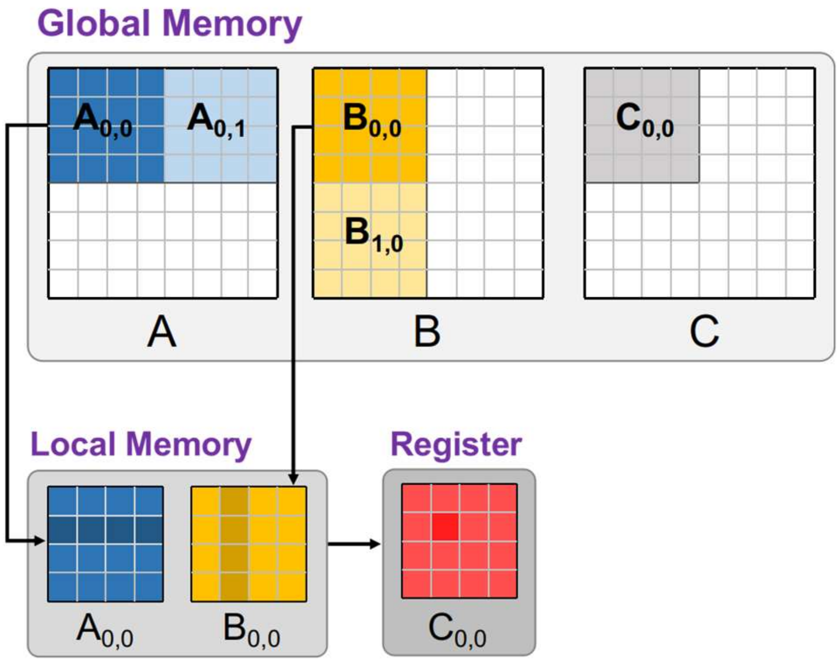
GEMM #4: Using Local Memory



- Local memory is allocated per work group
- User-managed data caches
- Access to Local memory is much faster than global memory

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

GEMM #4: Using Local Memory




```
for (int k = 0; k < N; k += TILE_K) {
  // load A
  for (int m = 0; m < cm; ++m) {
    int tile_row = i_row + m * block_size_row;
    int tile_col = i_col;
    int row = tile_row + g_row * cm * block_size_row;
    int col = tile_col + k;
    Asub[tile_row][tile_col] = A[row * N + col];
  }

  // load B
  for (int p = 0; p < cp; ++p) {
    int tile_row = i_row;
    int tile_col = i_col + p * block_size_col;
    int row = tile_row + k;
    int col = g_col * cp * block_size_col + tile_col;
    Bsub[tile_row][tile_col] = B[row * P + col];
  }

  // wait all memory has been stored to Asub & Bsub
  item.barrier(sycl::access::fence_space::local_space);

  for (int k1 = 0; k1 < TILE_K; ++k1) {
    for (int m = 0; m < cm; ++m) {
      for (int p = 0; p < cp; ++p) {
        Csub[m][p] += Asub[m + i_row * cm][k1] * Bsub[k1][p + i_col * cp];
      }
    }
  }
  item.barrier(sycl::access::fence_space::local_space);
}
```

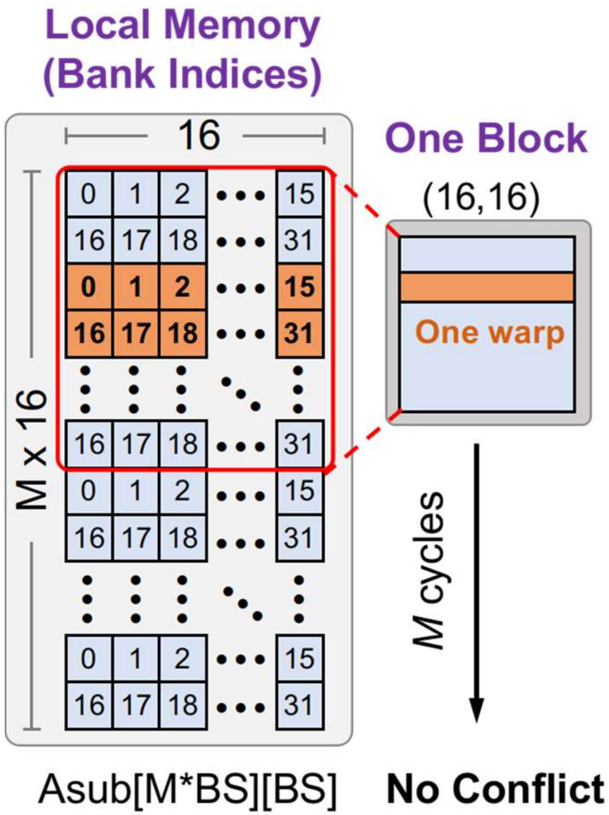
Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Performance

version	Gflops	Time (ms)	Step Speedup	Cumulative Speedup	Efficiency
v1	141.52	29.63	1.0x	1.0x	8.3%
v2	71.83	58.39	0.51x	0.51x	4.2%
v3	609.09	6.88	8.47x	4.31x	36%
v4	982.14	4.27	1.61x	6.96x	58%
Peak	1689	2.49	-	-	

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

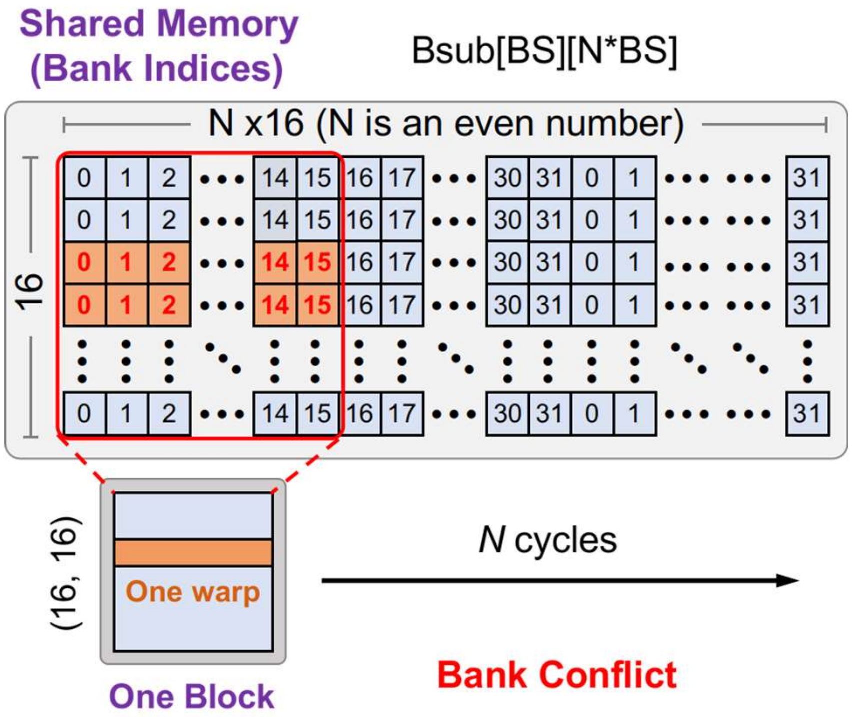
GEMM #5: Local Memory -- Bank Conflicts




```
for (int k = 0; k < N; k += TILE_K) {
  // load A
  for (int m = 0; m < cm; ++m) {
    int tile_row = i_row + m * block_size_row;
    int tile_col = i_col;
    int row = tile_row + g_row * cm * block_size_row;
    int col = tile_col + k;
    Asub[tile_row][tile_col] = A[row * N + col];
  }

  // load B
  for (int p = 0; p < cp; ++p) {
    int tile_row = i_row;
    int tile_col = i_col + p * block_size_col;
    int row = tile_row + k;
    int col = g_col * cp * block_size_col + tile_col;
    Bsub[tile_row][tile_col] = B[row * P + col];
  }

  // wait all memory has been stored to Asub & Bsub
  item.barrier(sycl::access::fence_space::local_space);
```

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

GEMM #5: Local Memory -- Bank Conflicts



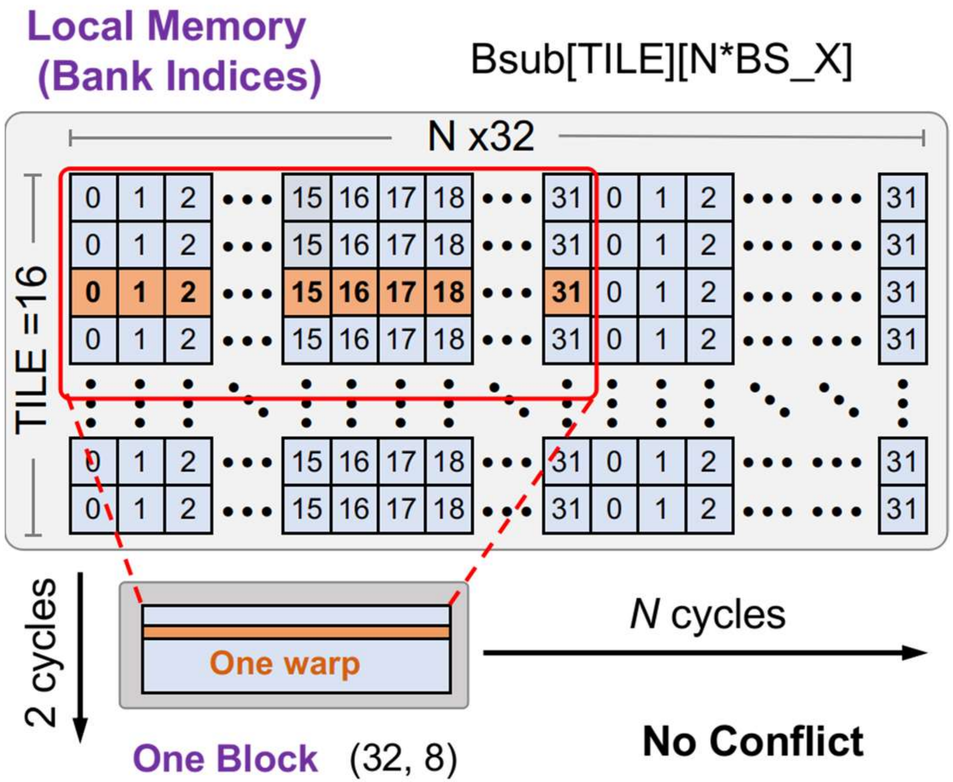

```
for (int k = 0; k < N; k += TILE_K) {
  // load A
  for (int m = 0; m < cm; ++m) {
    int tile_row = i_row + m * block_size_row;
    int tile_col = i_col;
    int row = tile_row + g_row * cm * block_size_row;
    int col = tile_col + k;
    Asub[tile_row][tile_col] = A[row * N + col];
  }

  // load B
  for (int p = 0; p < cp; ++p) {
    int tile_row = i_row;
    int tile_col = i_col + p * block_size_col;
    int row = tile_row + k;
    int col = g_col * cp * block_size_col + tile_col;
    Bsub[tile_row][tile_col] = B[row * P + col];
  }

  // wait all memory has been stored to Asub & Bsub
  item.barrier(sycl::access::fence_space::local_space);
```

- Different threads in one warp access "different" words in the same bank.

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

GEMM #5: Local Memory -- Bank Conflicts free

`#define BLOCK_SIZE_X 16` → `#define BLOCK_SIZE_X 32`



Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Performance

version	Gflops	Time (ms)	Step Speedup	Cumulative Speedup	Efficiency
v1	141.52	29.63	1.0x	1.0x	8.3%
v2	71.83	58.39	0.51x	0.51x	4.2%
v3	609.09	6.88	8.47x	4.31x	36%
v4	982.14	4.27	1.61x	6.96x	58%
v5	1048.73	3.99	1.06x	7.43x	62%
Peak	1689	2.49	-	-	

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

What to do Next?

- At v5 1048GFLOP/s, we’re 62% Efficency.
- Next Step
  - Vectorization: using float4
  - Reduce instruction overhead: unroll loops
  - Tuning
- Trying ⇄ Finding Cause ⇄ Proving

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Conclusion

- Use peak performance metrics to guide optimization
- Understand performance characteristics
    - Latency hiding
    - Bank conflict
- Try to identify the type of bottleneck
    - Using profiling tool: vtune
    - Memory, core computation, or instruction overhead
- Learn more about Hardware
- Look into the assembly code

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Thank You！