Fast Convolution Operations on Many-Core Architectures

Reporter: Shigang Li
Affiliation: SKL of Computer Architecture
Institute of Computing Technology, CAS
Email: shigangli.cs@gmail.com

17th International Conference on High Performance Computing and Communications (HPCC'15)

Outline

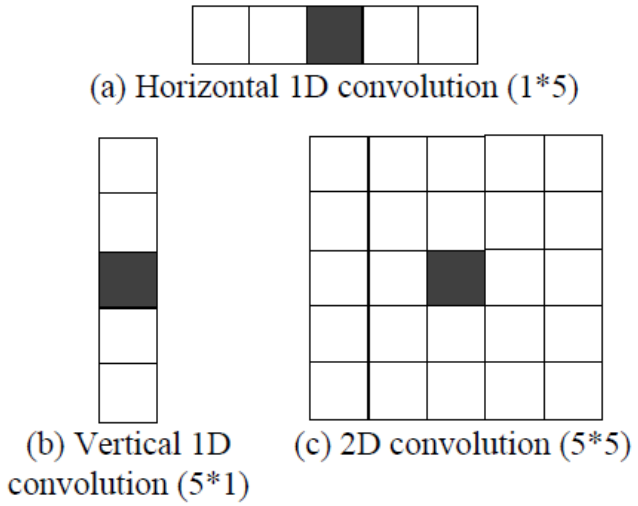
ICT, CAS

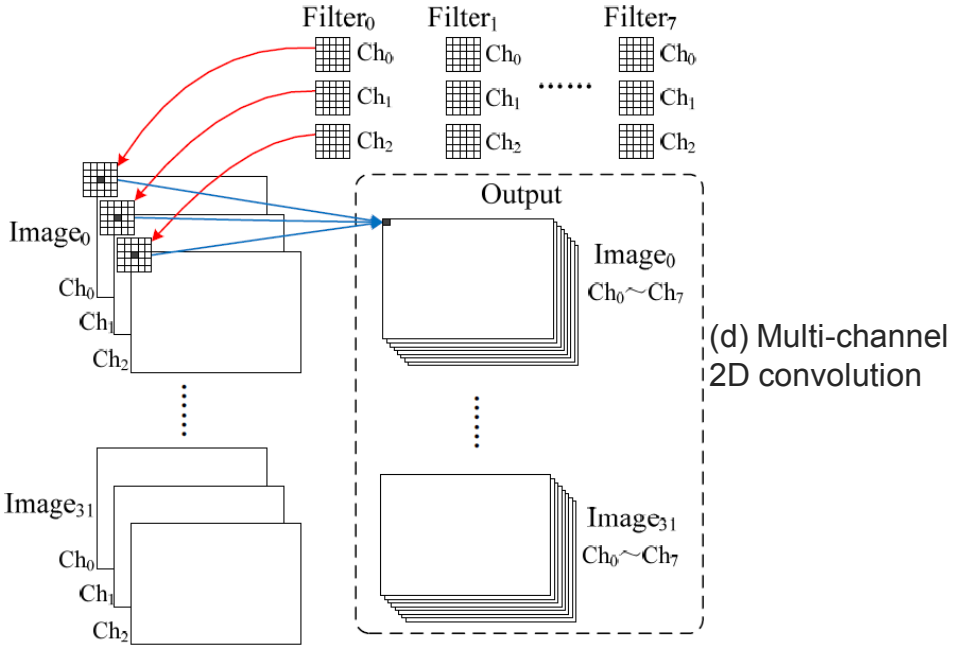
1. Introduction
2. 1D and 2D convolutions
3. Multi-channel 2D convolution
4. Experimental results
5. Conclusions

Different convolution operations

ICT, CAS

- In the context of image processing, convolution is the scalar product of the filter weights with the input pixels within a window surrounding each of the output pixels, such as 1D, 2D and multi-channel 2D convolutions.





- They are widely used in image processing, such as Gaussian filter; today's Deep Learning, such as Convolution Neural Network.

Many-core processors bring opportunity

- Convolution operations have high computational and memory access complexity.
- Many-core architectures evolve quickly, which feature high computational throughput and memory bandwidth, like NVIDIA and AMD GPUs, and Intel Xeon Phi.
- This paper aims at providing some insights for performance optimizations of convolution operations on these emerging many-core architectures.

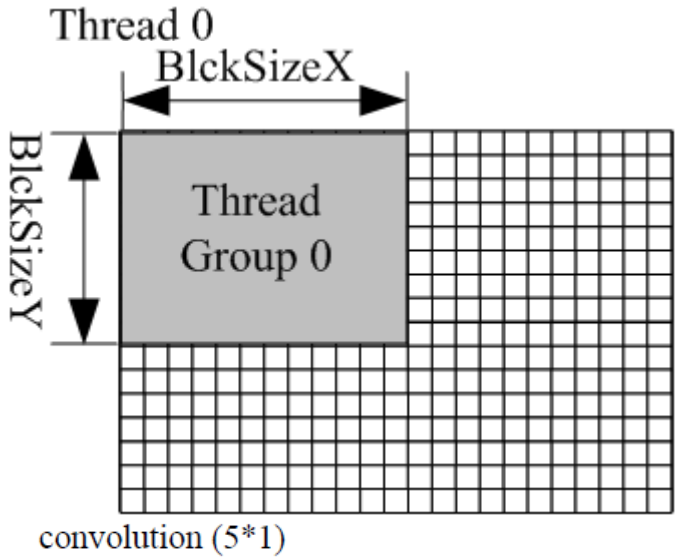
Outline

ICT, CAS

1. Introduction
2. 1D and 2D convolutions
3. Multi-channel 2D convolution
4. Experimental results
5. Conclusions

1D and 2D convolutions

ICT, CAS



We discuss the optimizations in the context of OpenCL:

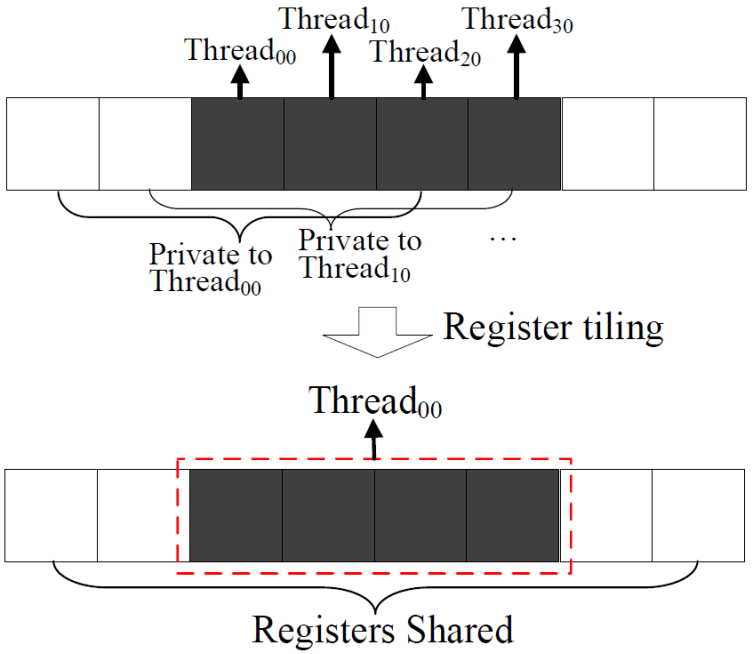
1. A naïve implementation of 1D and 2D convolutions is each thread is responsible for one module and accesses input pixels from global memory directly.

2. A common optimization is utilizing local memory tiling to cache the input pixels, which reduces the total global memory accesses.

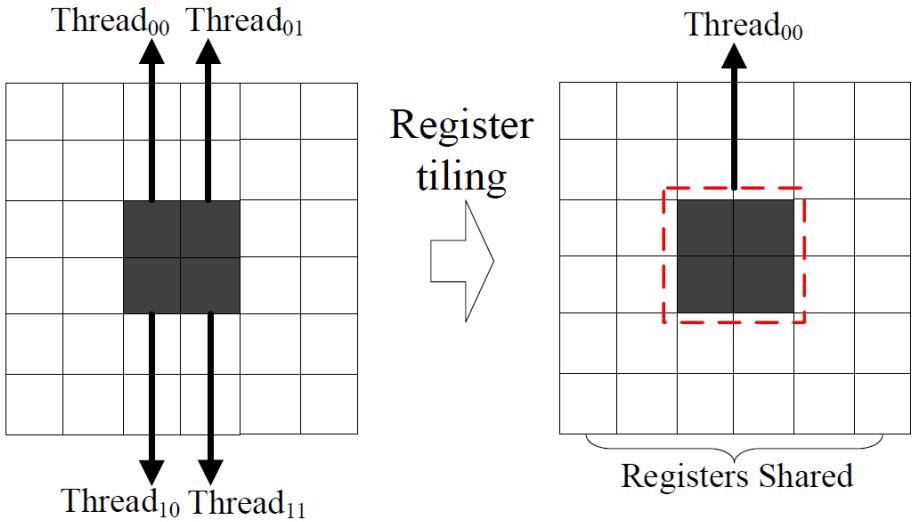
Register tiling for 1D and 2D convolutions

ICT, CAS

In order to exploit the register-level data reuse, register tiling by coarsening the workload of each thread is used.



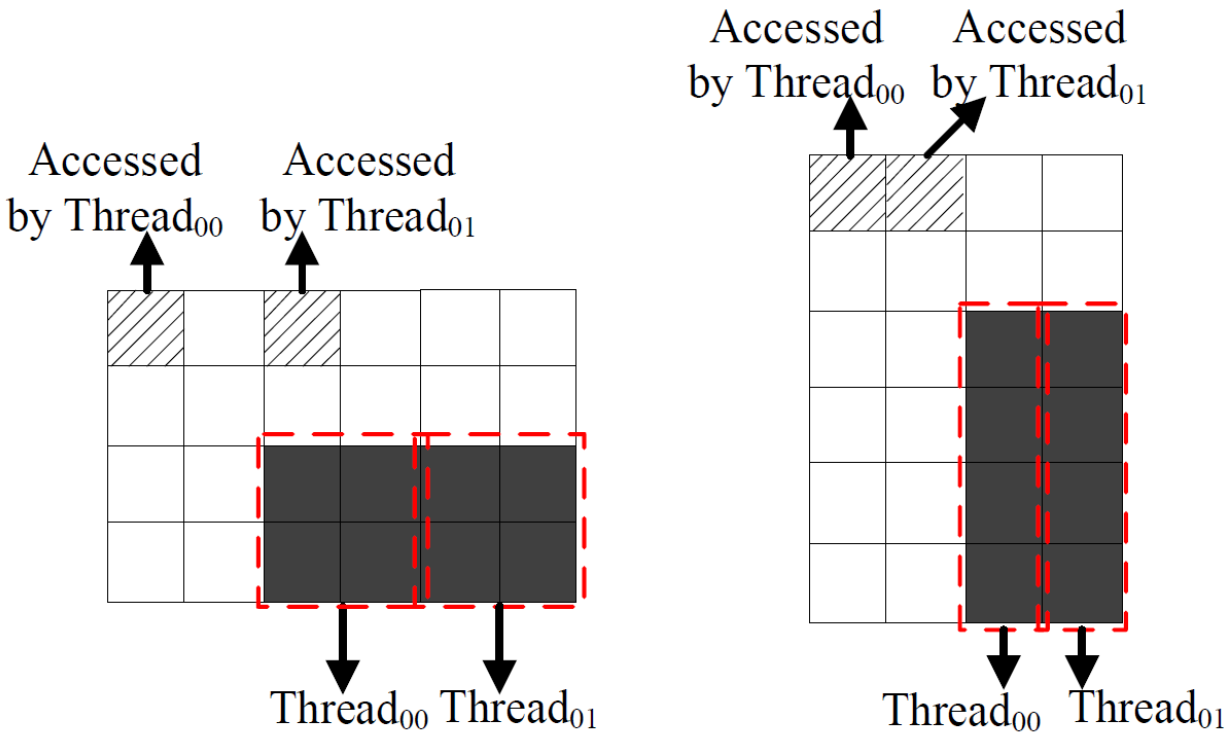
(a) Register tiling for horizontal 1D convolution



(b) Register tiling for 2D convolution

Register tiling affects memory efficiency

ICT, CAS



(a) Register tiling 2*2

(b) Register tiling 4*1

Figure (a) is not coalesced memory access.

Figure (b) is coalesced memory access.

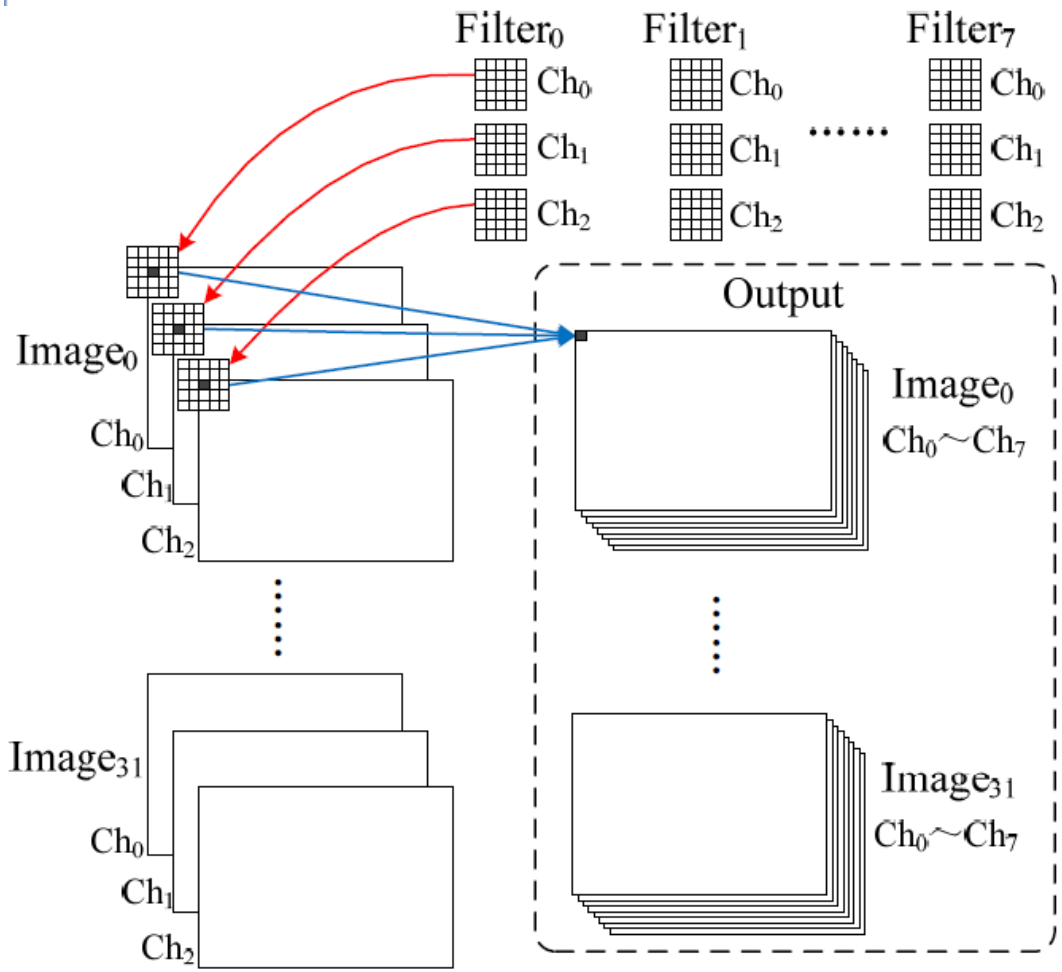
Outline

ICT, CAS

1. Introduction
2. 1D and 2D convolutions
3. Multi-channel 2D convolution
4. Experimental results
5. Conclusions

Multi-channel 2D convolution in mini-batch mode

ICT, CAS



Basically, the serial algorithm contains total 7 nested *for* loops.

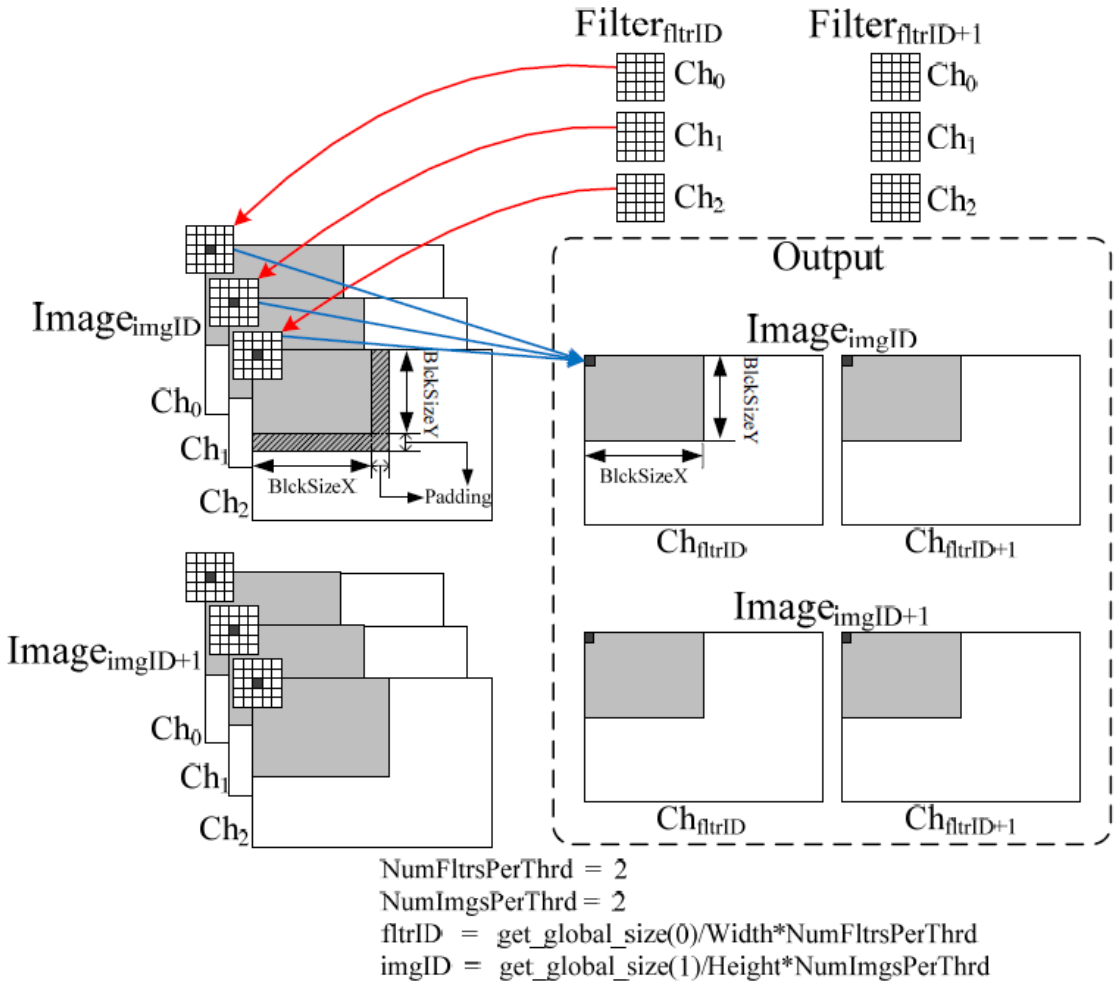
The outermost 2-level *for* loops iterate on NumImages (i.e. batchsize) and NumFilters (number of filters), respectively.

The inner 5-level for loops, from outside to inside, iterate on image Height and Width, Channels, FilterSize and FilterSize, respectively

A naive method to implement this algorithm in OpenCL is to let each thread calculate for one pixel of the output images.

Optimization details at thread-group level

ICT, CAS



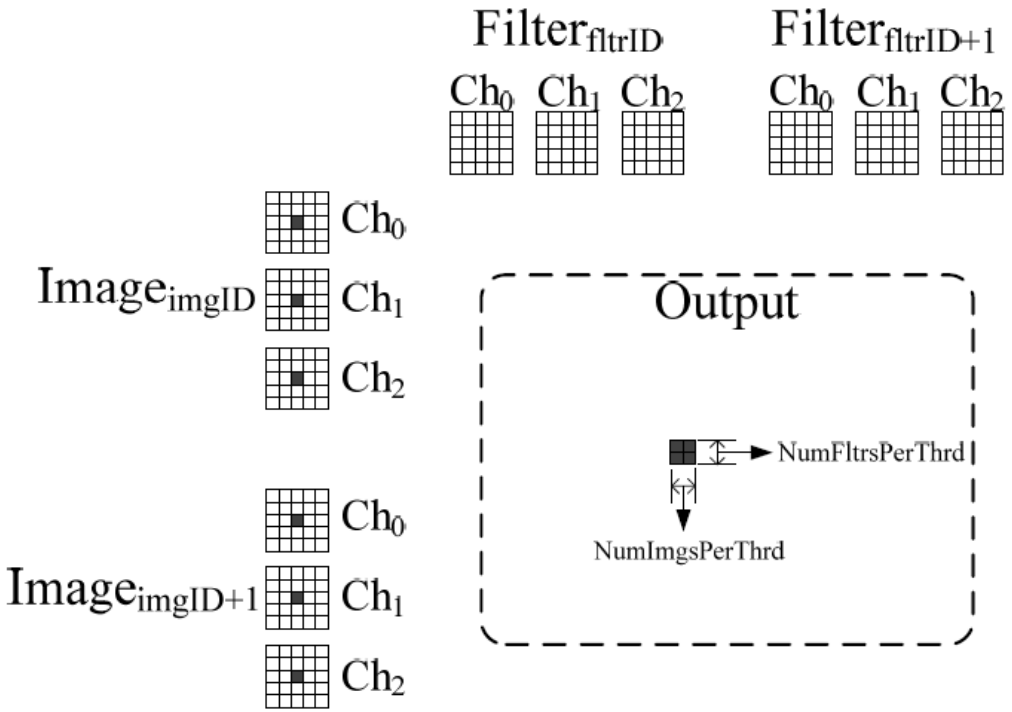
1. Each threads group is responsible for calculating NumFltrsPerThrd* NumImgsPerThrd blocks of the output images, which exploits the data reuse of the outermost 2-level *for* loops, i.e. loop tiling.

2. Each block of input image is loaded into local memory before calculating, i.e. local memory tiling, which exploits inter-module data reuse.

3. Filters are put in constant memory to utilize constant cache.

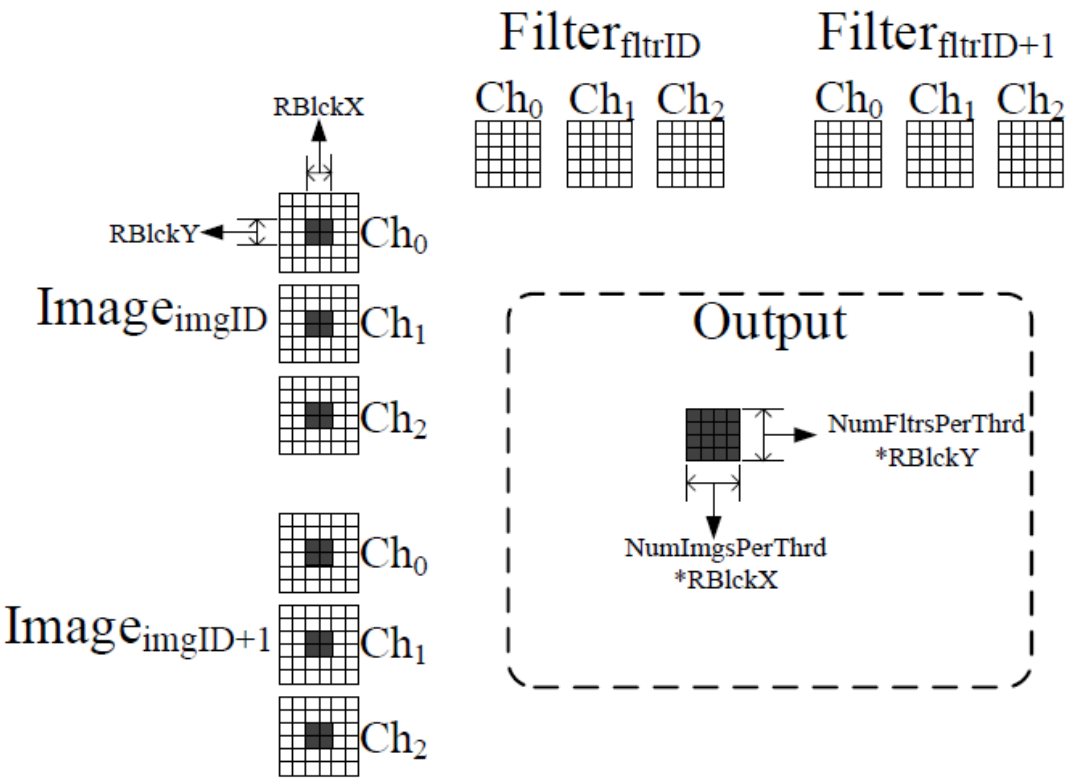
Optimization details at thread level

ICT, CAS



For each channel, each thread streams in the pixels of NumImgsPerThrd images and the values of NumFltrsPerThrd filters to registers, and then execute the convolution on the current channel.

Register tiling for Multi-channel 2D convolution



Register tiling can also be used to further exploit the inter-module date reuse. As we can see in the figure, the workload of each thread is coarsened.

Outline

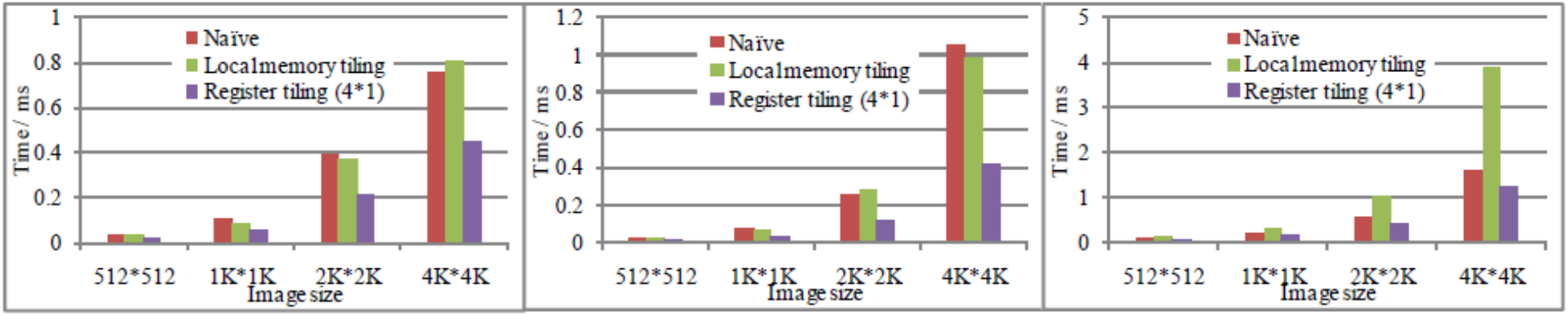
ICT, CAS

1. Introduction
2. 1D and 2D convolutions
3. Multi-channel 2D convolution
4. Experimental results
5. Conclusions

Performance results of 1D convolution

ICT, CAS

All of the codes are implemented in OpenCL. We use AMD W8000 (3.23TFlops), GTX TITAN (4.5TFlops) and Intel MIC (2.0496TFlops to do the experimental analysis.



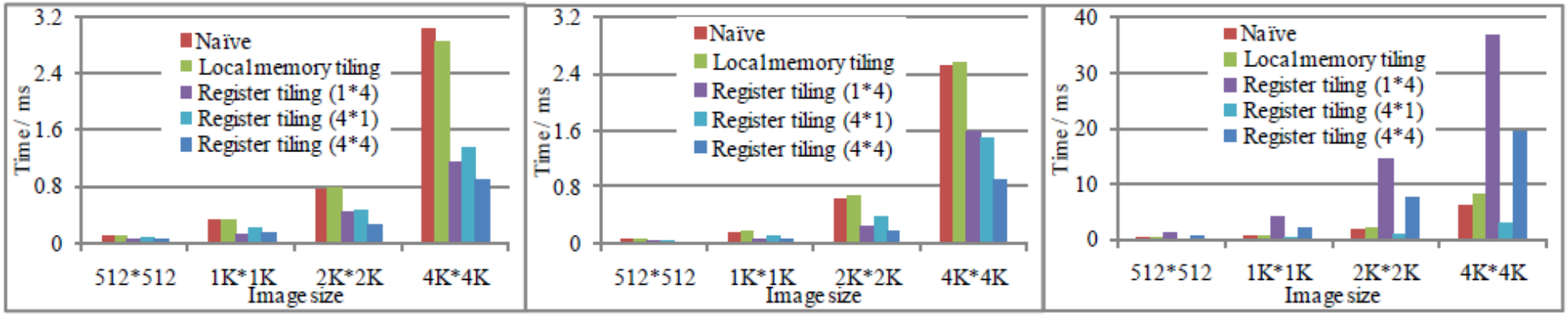
(a) Vertical 1D convolution on AMD W8000 (b) Vertical 1D convolution on GTX TITAN (c) Vertical 1D convolution on Intel MIC

Horizontal 1D convolution

1. On AMD W8000 and GTX TITAN, the performance of the naive method and local memory tiling is almost equal. This is because that the naïve method may benefit from hardware cache.
2. Register tiling greatly improves the performance for both horizontal and vertical 1D convolutions on AMD W8000 and GTX TITAN.
3. On Intel MIC, local memory tiling performs much worse, as no real hardware support for local memory.

2D convolution

ICT, CAS



(a) 2D convolution on AMD W8000 (b) 2D convolution on GTX TITAN (c) 2D convolution on Intel MIC

2D convolution

1. Register tiling, with tiling size of 4*4, usually performs best for AMD W8000 and GTX TITAN.

2. On Intel MIC, register tiling along the vertical dimension with tiling size of 4*1, which guarantees consecutive memory access, always performs best.

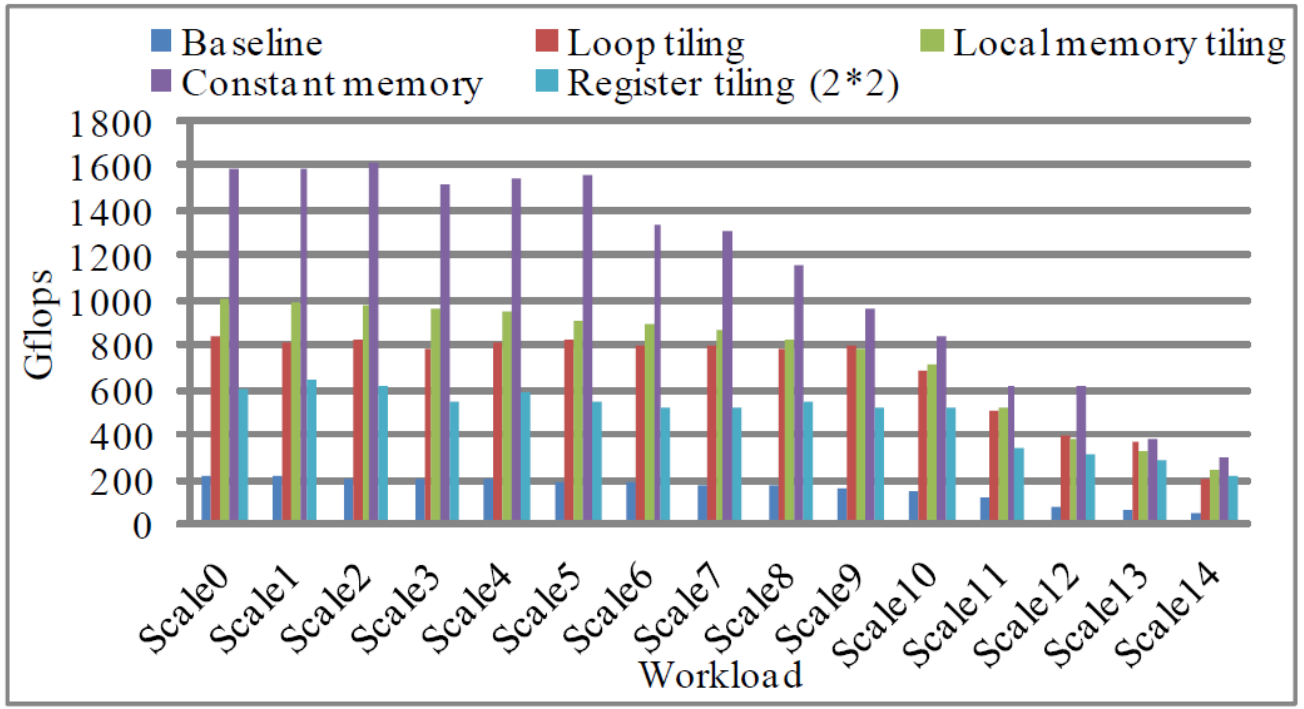
Test scales of convolution layer of CNN

ICT, CAS

	image size	batch size	channels	filters	filter size
$Scale_0$	128*128	128	3	96	13*13
$Scale_1$	128*128	64	8	96	13*13
$Scale_2$	128*128	64	16	96	13*13
$Scale_3$	128*128	128	3	96	11*11
$Scale_4$	128*128	64	8	96	11*11
$Scale_5$	128*128	64	16	96	11*11
$Scale_6$	128*128	128	96	128	9*9
$Scale_7$	64*64	64	96	128	9*9
$Scale_8$	32*32	64	96	128	9*9
$Scale_9$	64*64	128	128	128	7*7
$Scale_{10}$	32*32	64	128	128	7*7
$Scale_{11}$	16*16	64	128	128	7*7
$Scale_{12}$	32*32	128	128	384	3*3
$Scale_{13}$	16*16	64	128	384	3*3
$Scale_{14}$	13*13	64	128	384	3*3

Optimizations comparison of multi-channel 2D convolution

ICT, CAS

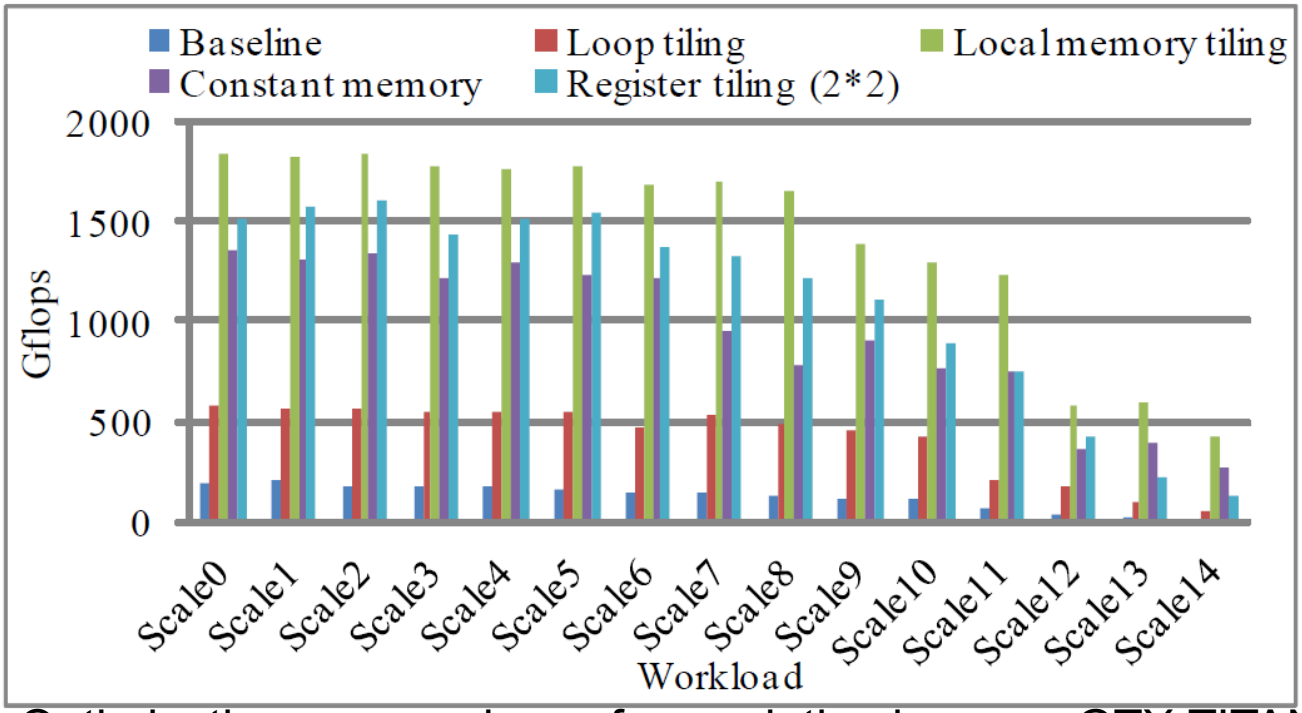


Optimizations comparison of convolution layer on AMD W8000

Using constant memory for filters improves the performance greatly on AMD W8000.

Optimizations comparison of multi-channel 2D convolution

ICT, CAS

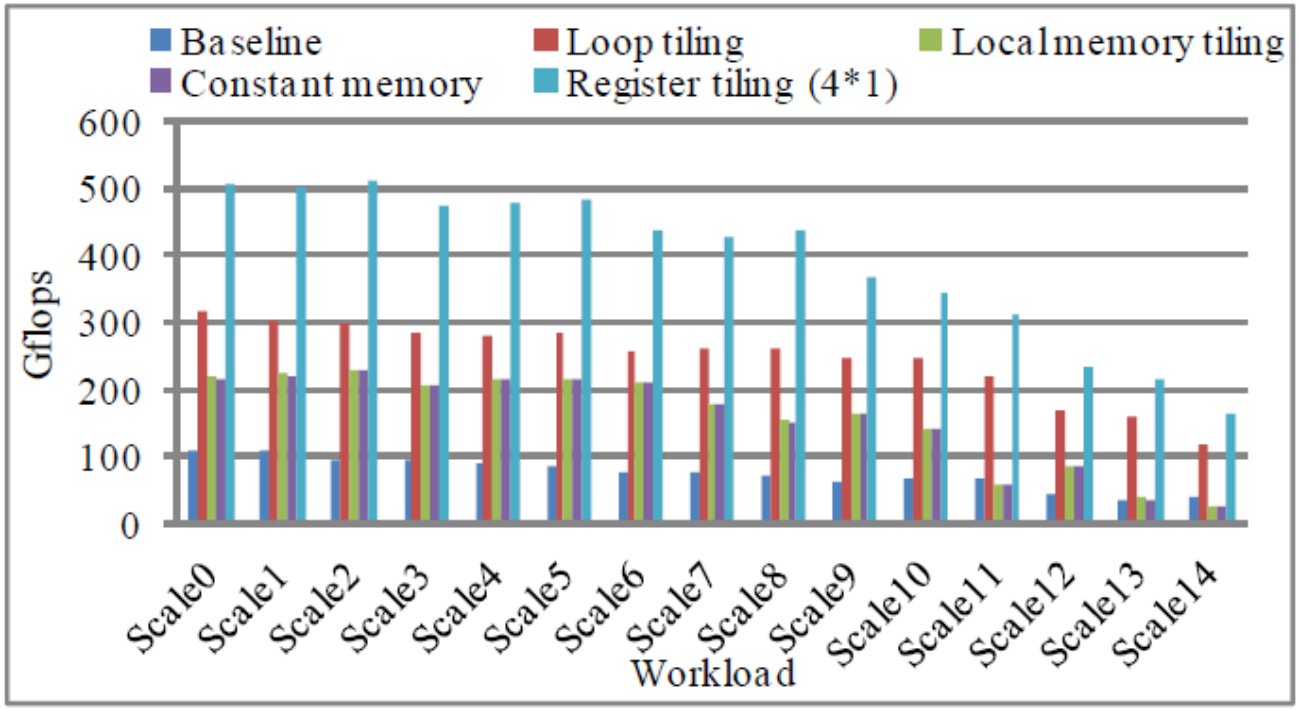


Optimizations comparison of convolution layer on GTX TITAN

We can see that the best performance is obtained after local memory tiling. However, the constant memory defined in OpenCL doesn't work for NVIDIA GPUs.

Optimizations comparison of multi-channel 2D convolution

ICT, CAS

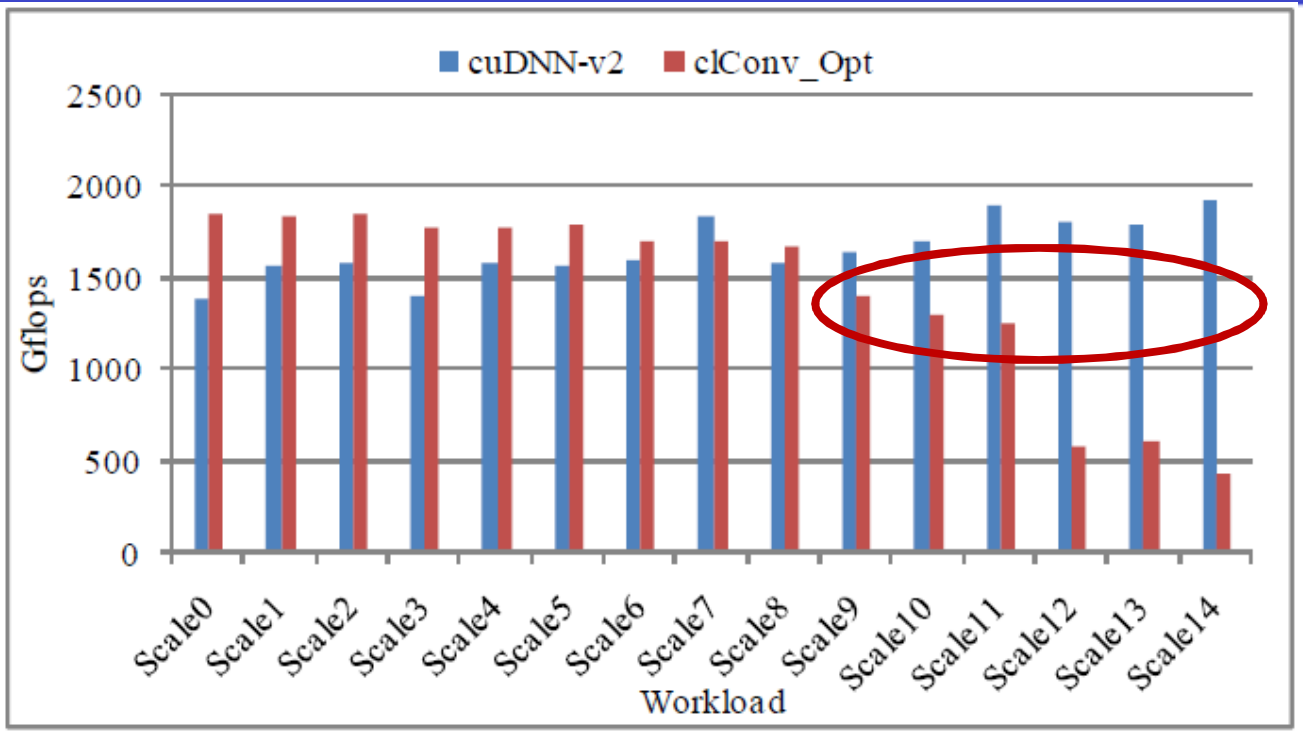


Optimizations comparison of convolution layer on Intel MIC

The best performance is obtained when the register tiling is conducted only along the vertical dimension, which guarantees consecutive memory access. However, local memory tiling and constant memory for filters don't work on Intel MIC.

Compared with cuDNN-v2

ICT, CAS



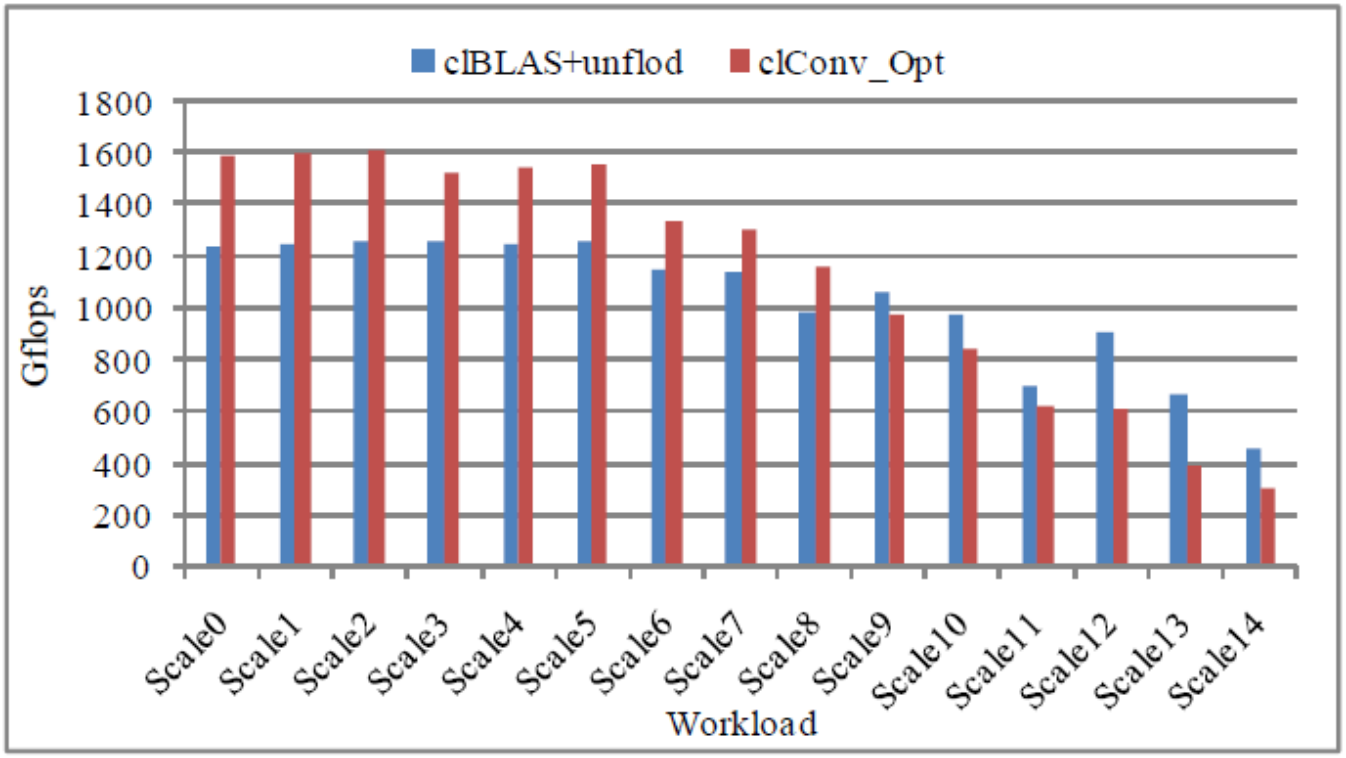
Performance comparison with cuDNN-v2 on GTX TITAN

1. For the large filter size, our solution performs apparently better than cuDNN-v2, with up to 33% performance improvement. Because our solution fully exploits inter-module date reuse.

2. However, our solution performs worse than cuDNN-v2 when the filter size is small, which is caused by the low inter-module data reuse rate of small filter size.

Compared with clBLAS+unfold

ICT, CAS



Compared with clBLAS+unfold on AMD W8000

clBLAS+unfold follows the same method with the well-known deep learning library—CAFFE.

1. Our solution performs apparently better than clBLAS+unfold for large filter size, with up to 28% performance improvement.

2. However, our solution performs worse when the filter size is small.

Outline

ICT, CAS

1. Introduction
2. 1D and 2D convolutions
3. Multi-channel 2D convolution
4. Experimental results
5. Conclusions

Conclusions and future work

ICT, CAS

- For 1D and 2D convolutions, register tiling rather than local memory tiling is critical to achieve good performance. And different register tiling modes affect the memory access efficiency.

- Our solution for multi-channel 2D convolution fully exploits inter-module data reuse, and thus gets good performance for large filter sizes, up to 33% over cuDNN-v2 and up to 28% over clBLAS.

- To improve the performance of multi-channel 2D convolution for small filter size is our future work.

Q&A

ICT, CAS

Thank You

Reporter: Shigang Li

Email: shigangli.cs@gmail.com