INTERNATIONAL CONFERENCE ON PARALLEL INTERNATIONAL

Optimizing Massively Parallel Winograd Convolution on ARM Processor

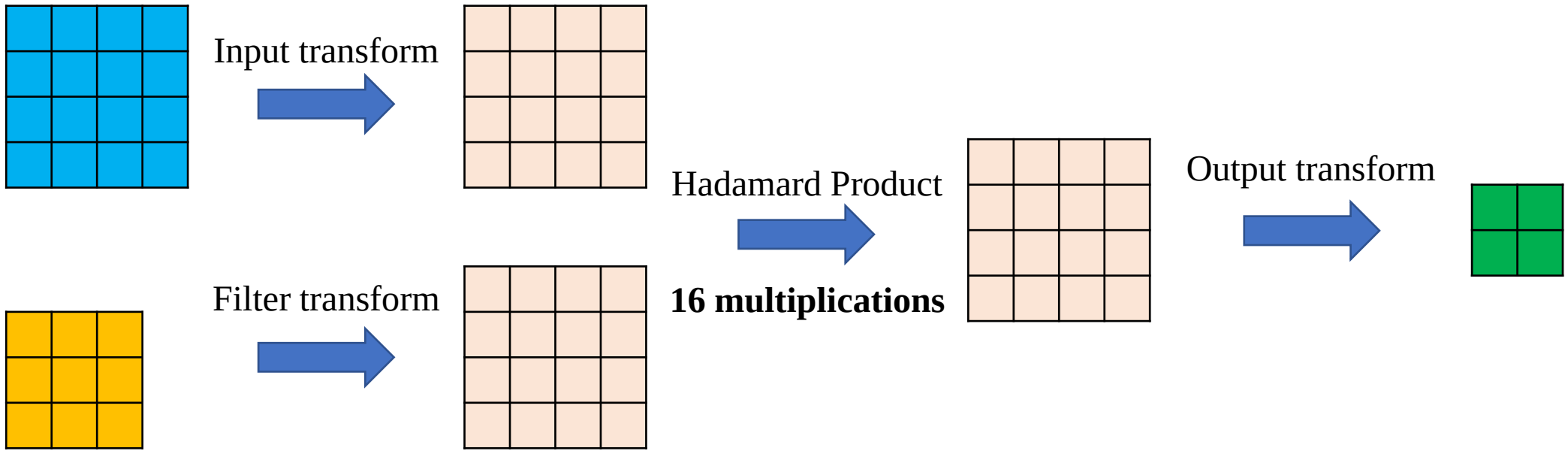
Dongsheng Li， Dan Huang, Zhiguang Chen, Yutong Lu

Sun Yat-sen University

August, 9-12, 2021

Winograd convolution

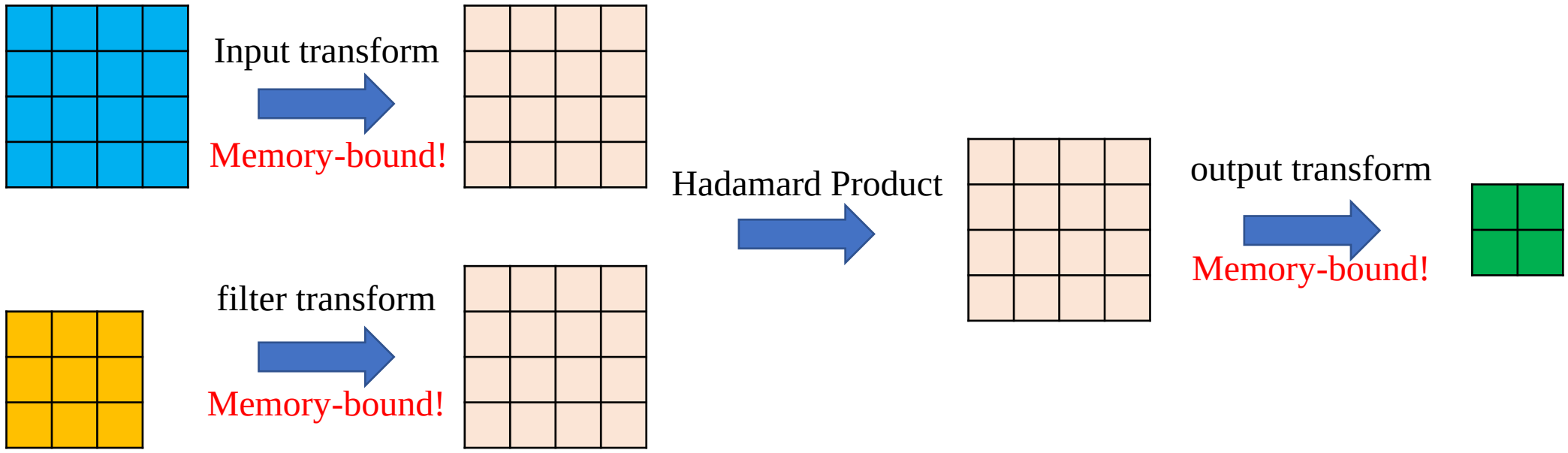
Convolution Neural Network (CNN) has gained a great success in deep learning applications, such as computer vision and natural language processing.



Winograd convolution can eliminate the multiplications for convolutional layers. Compared to direct convolution, it can achieve 36 / 16 = 2.25x speedup!

Bottleneck

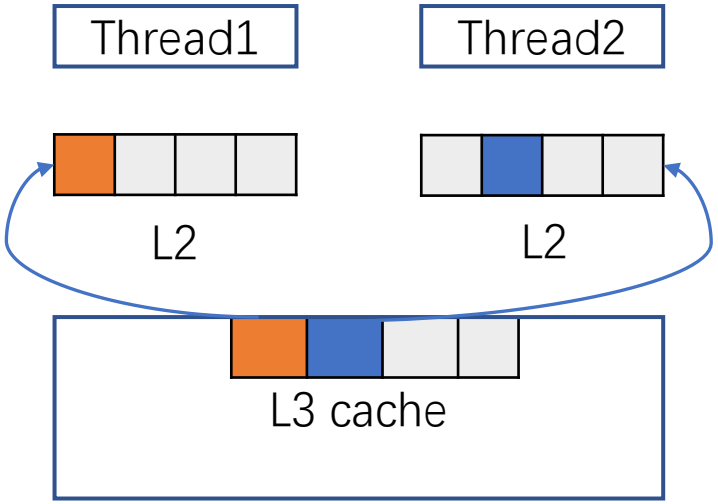
Winograd convolution can eliminate the multiplications for convolutional layers.



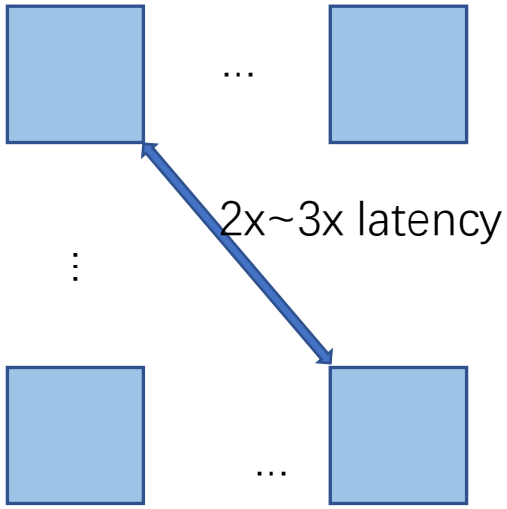
However, transform overhead will affect the practical performance, which may cost more than the time saved.

Bottleneck on ARM manycore processor

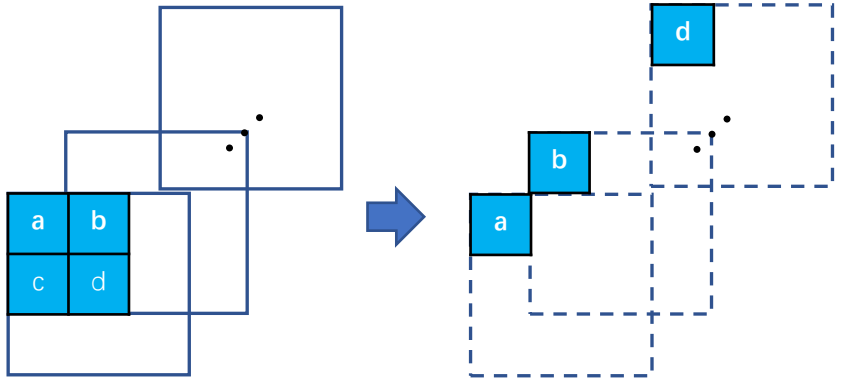
The performance becomes worse on ARM manycore processor…



Cache contention



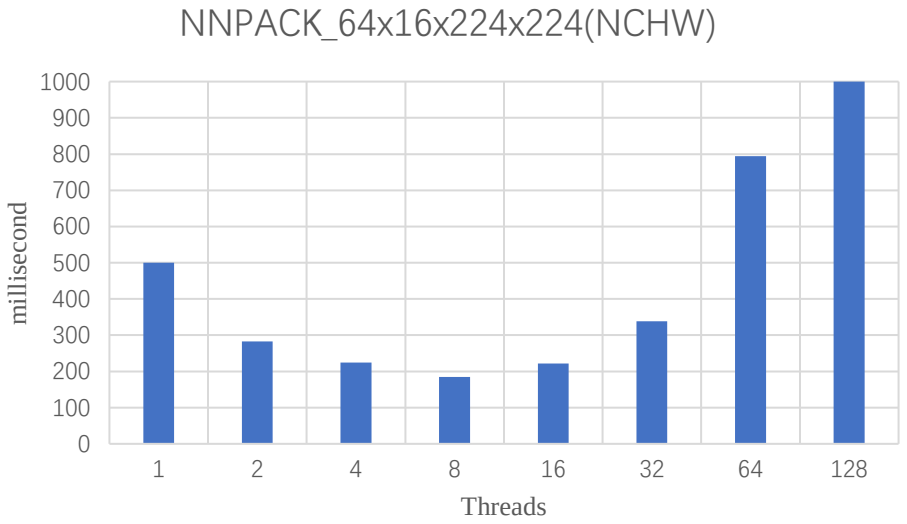
Remote access and memory congestion



Non-sequential memory access

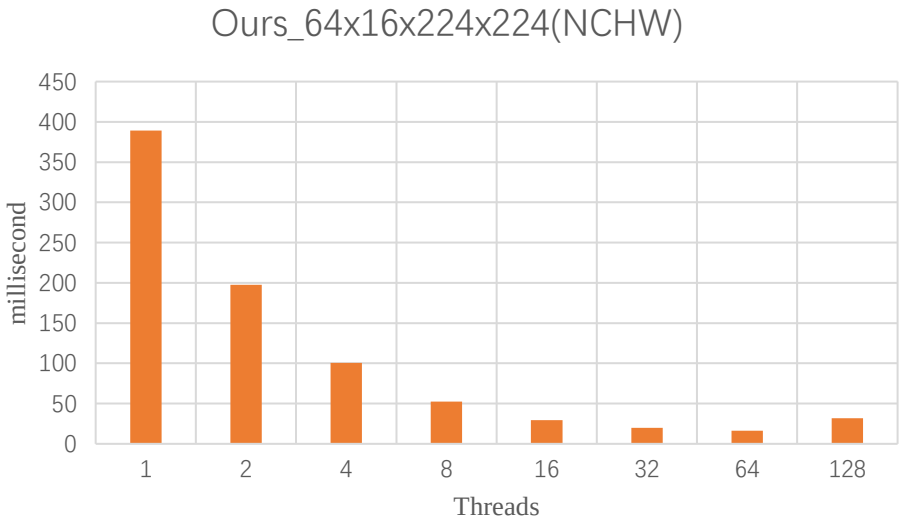
Parallel performance on ARM manycore processor

Parallel performance for input image size of 64x16x224x224 (NCHW) and filter size of 16x16x3x3.



NNPACK

VS.

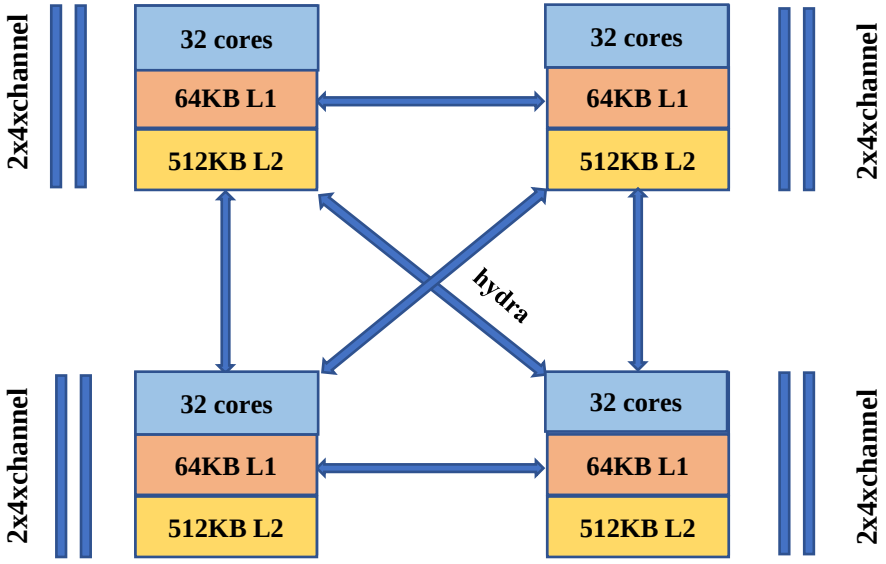


Ours

* NNPACK is an acceleration package for neural network computations.

ARM processor

- Challenge
  - Smaller L3 cache per core
  - Higher bandwidth pressure
  - More remote access
  - More memory congestion



Core	
number	128
name	TSv110
frequency	2.6GHZ
die count	4

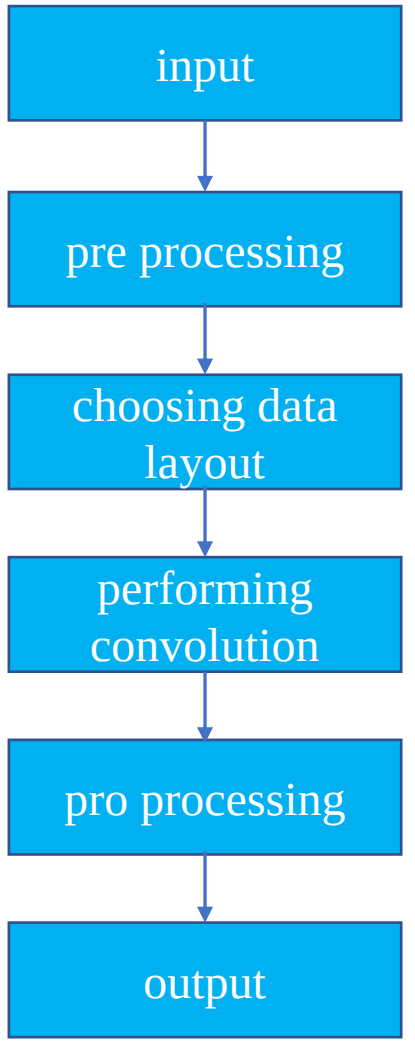
Memory	
Type	DDR4
channel	8
NUMA node	4

Cache	
L1 I	64KB x 128
L1 D	64KB x 128
L2	512KB x 128
L3	16MB x 4
L2 cache line	64B
L3 cache line	128B

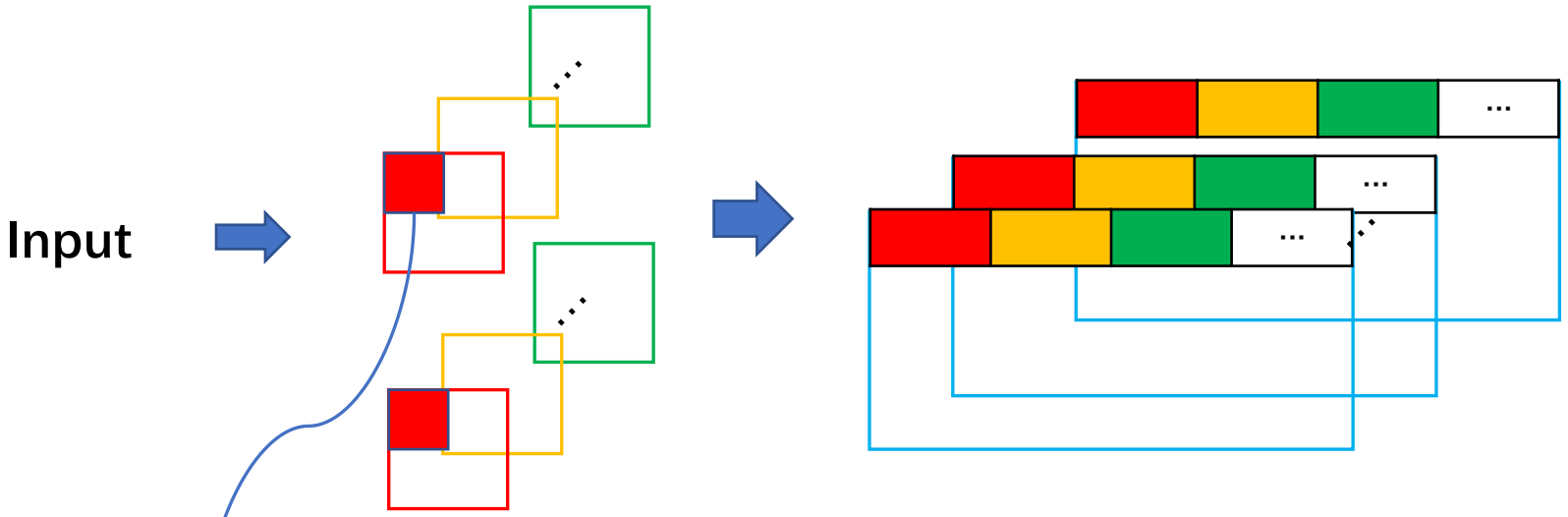
Features	
SIMD	neon

Overview

- **Reduce non-sequential memory access and memory copy operations**
  - Reorder data layout in transformation stage
  - Minimize padding overhead
- **Achieve high hardware utilization**
  - Choosing data layout according to input tensor shape
  - Utilizing GEMM routine and optimizing computation for reordered layout
  - Expose enough parameters to adapt different hardware
- **Using divide-and-conquer algorithm to construct data parallelism**
  - Recursively breaks down a problem into more sub-problems
  - Provide a disjoint aligned buffer for each core
  - NUMA-scheduler to schedule data to corresponding nodes



Input transformation

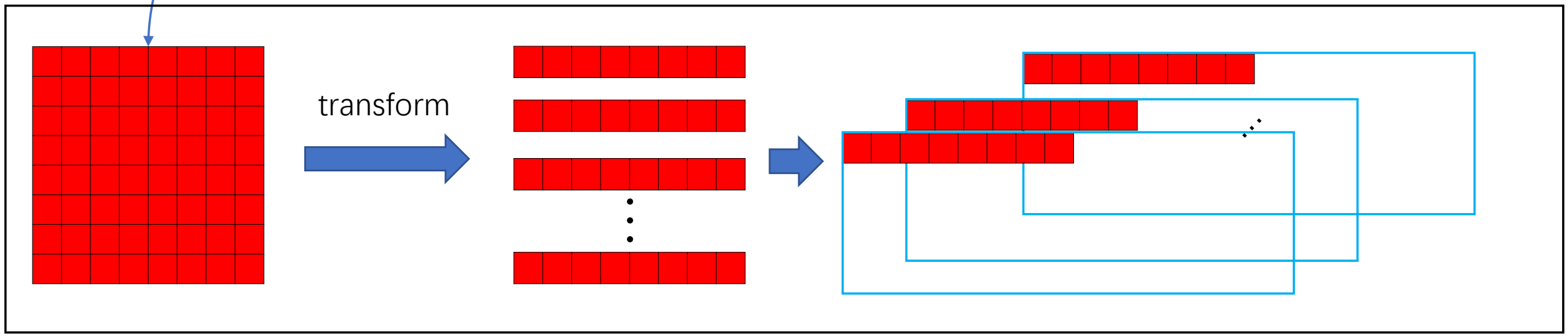


$$\begin{matrix} i \\ i+1 \end{matrix} \begin{bmatrix} \cdots & & & & & & & & \\ 0 & 1 & 1 & -17/4 & -17/4 & 1 & 1 & 0 \\ 0 & -1 & 1 & 17/4 & -17/4 & -1 & 1 & 0 \\ & & & & \cdots & & & & \end{bmatrix} * r$$

$$out[i]=r[1]+r[2]-17/4r[3]-17/4r[4]+r[5]+r[6]$$
$$out[i+1]=-r[1]+r[2]+17/4r[3]-17/4r[4]-r[5]+r[6]$$

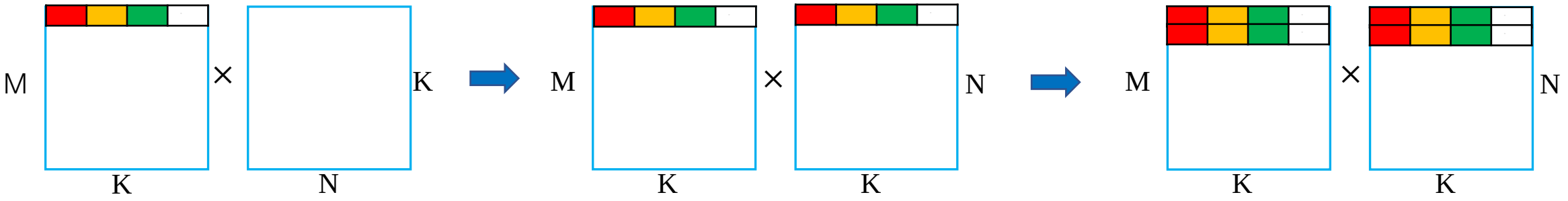
$$tmp0=(r[2]+r[6]-17/4r[4])$$
$$tmp1=(r[1]+r[5]-17/4r[3])$$
$$out[i]=tmp0+tmp1$$
$$out[i+1]=tmp0-tmp1$$

manually unrolling loop



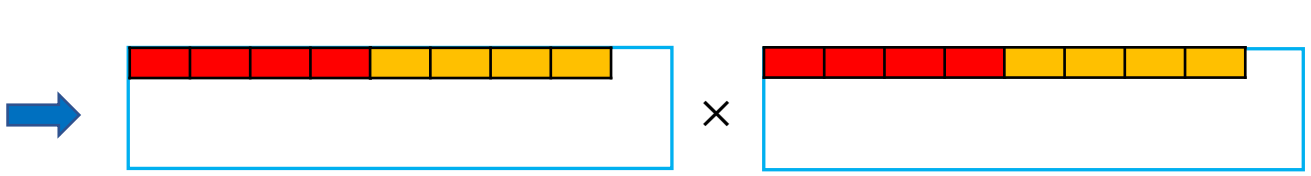
Optimization for reordered data layout

Through some techniques optimizing computing routine.

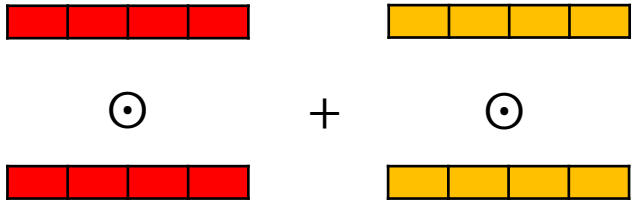


rearrange data layout

cache blocking



packing



register blocking and fma instruction

Choosing algorithm parameters

Choosing Parameters (Search or Testing)

- Vector length
  - Multiple of 4 (ARM vector length), e.g. 8.
- Cache blocking size
  - Adapting to L2 cache size.
- Register blocking size
  - Need 28 vector registers.

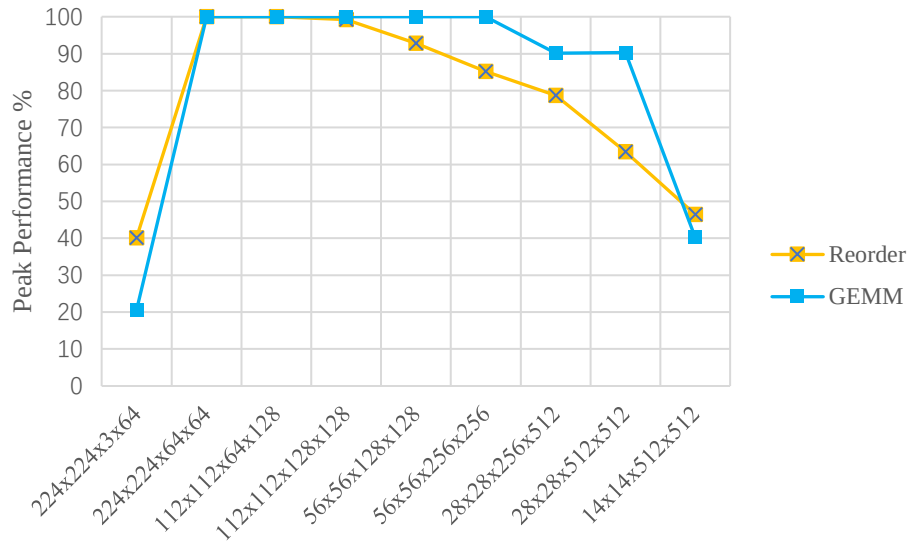
Choosing data layout

Data Layout (take VGG16 as an example)

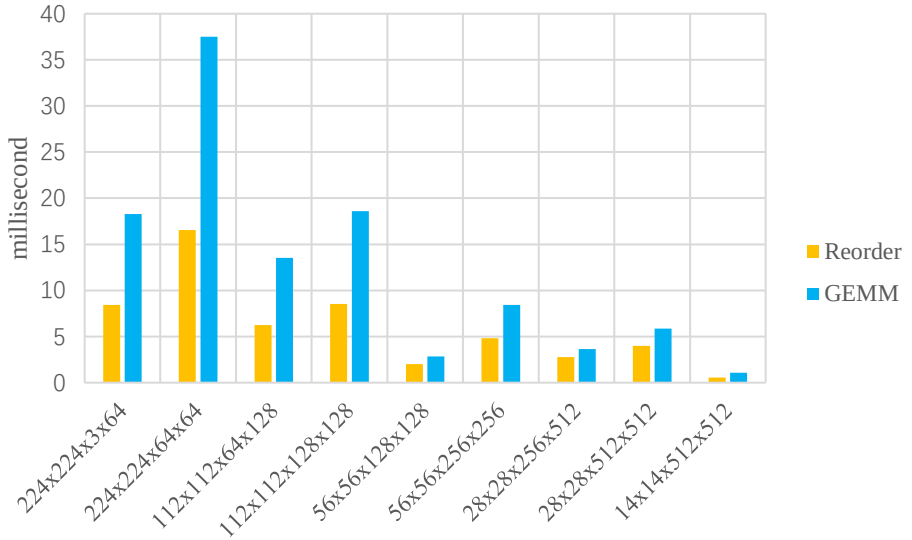
Large Input size
Small channel size

Small Input size
Large channel size

Peak performance at computation stage

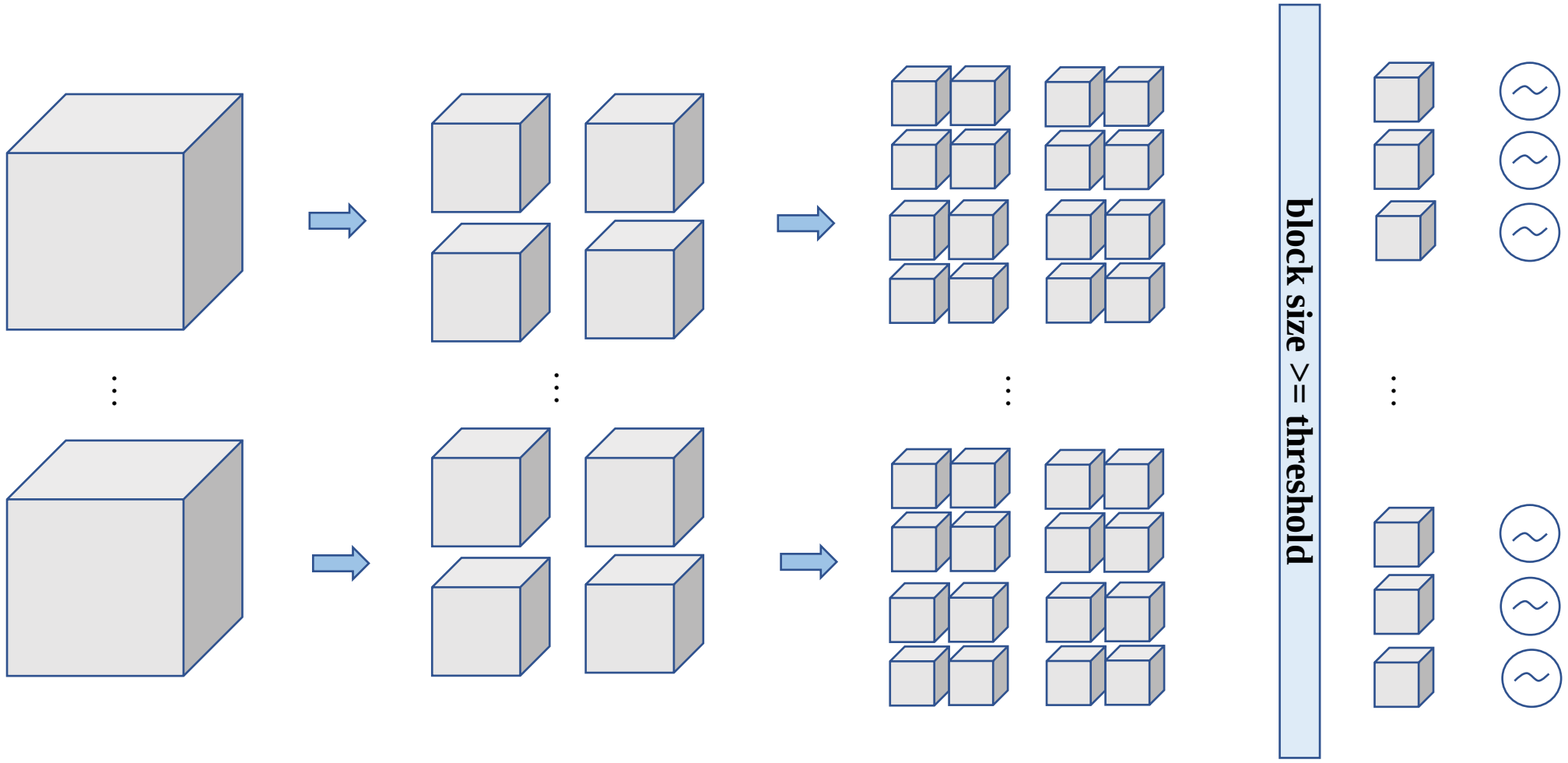


Transform overhead

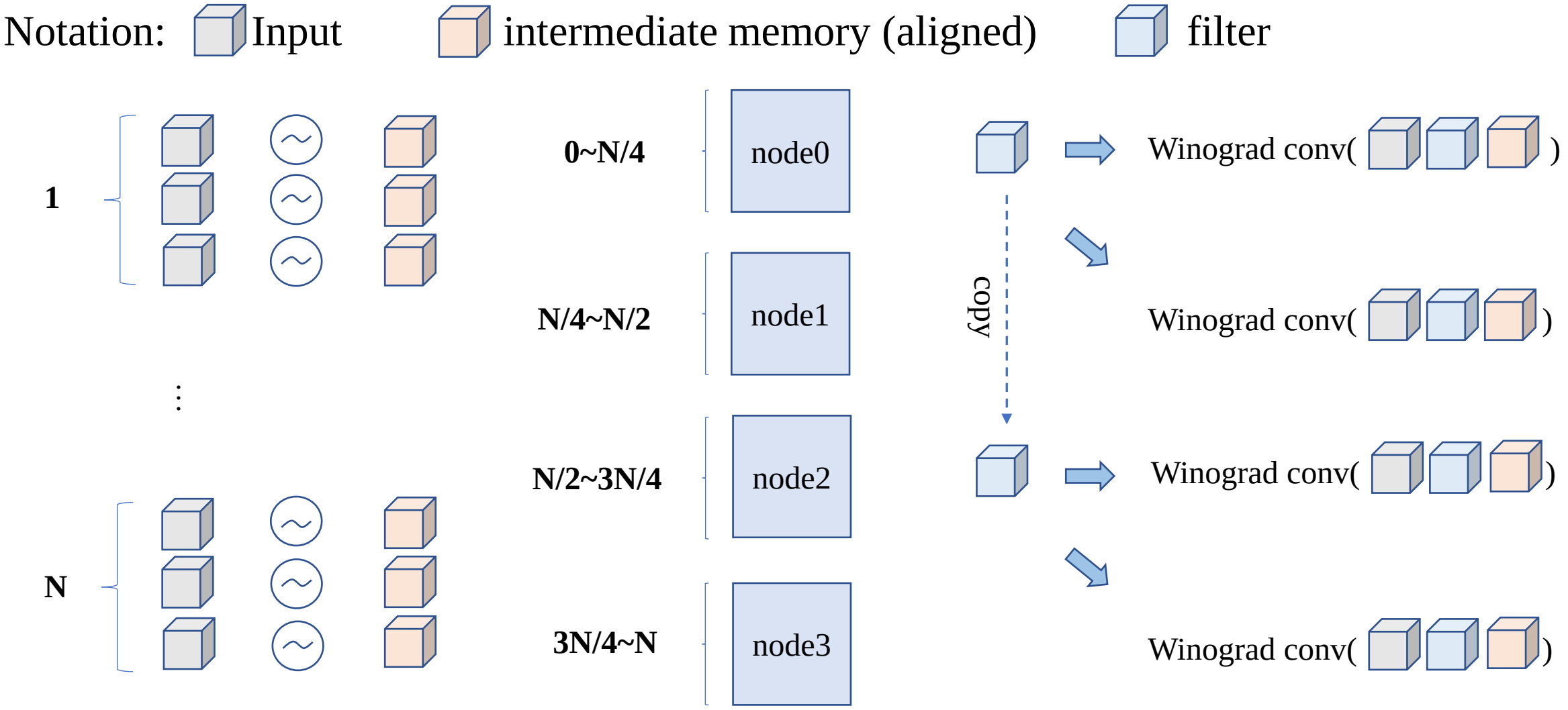


Parallelization

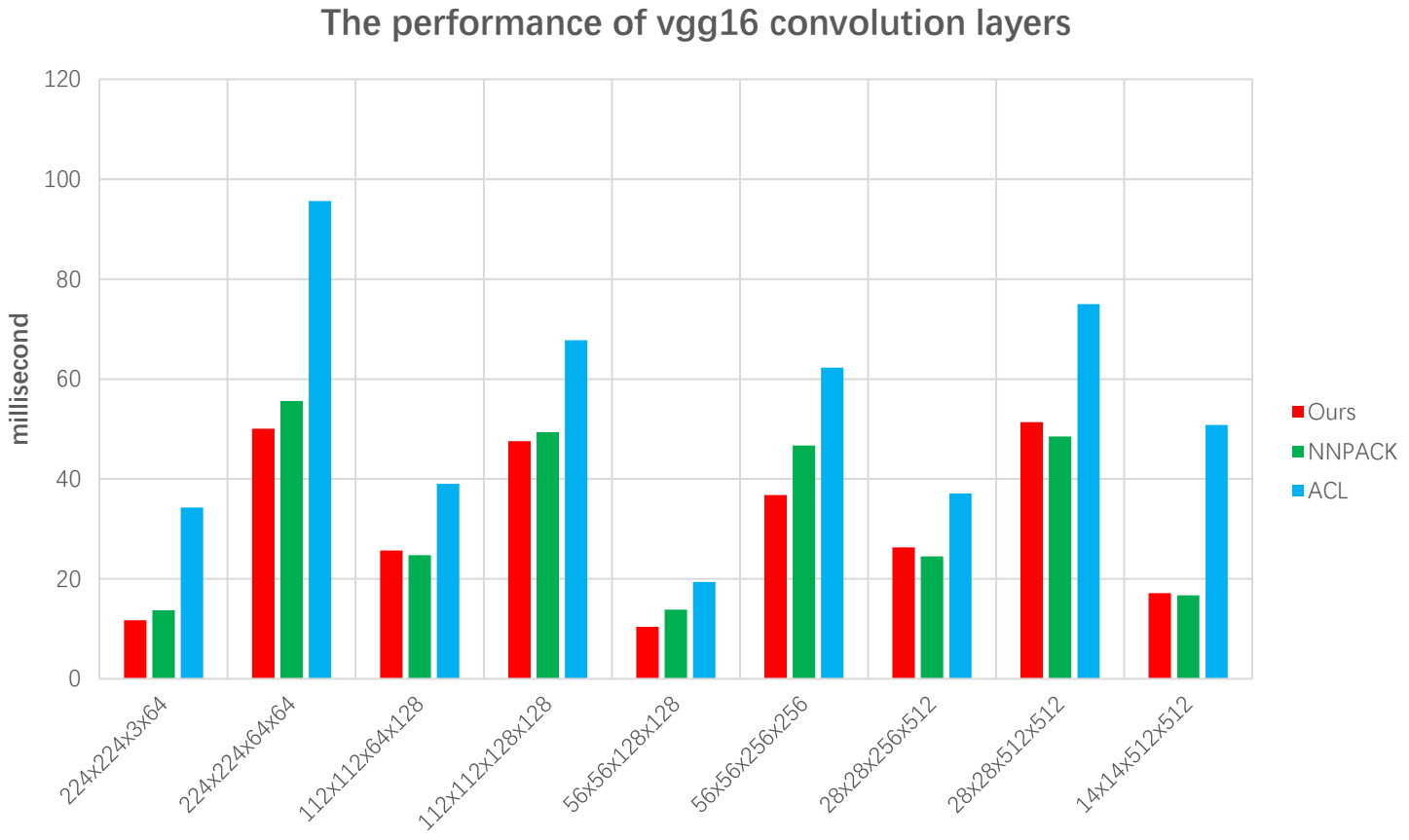
Divide-and-conquer parallel algorithm



NUMA Scheduling



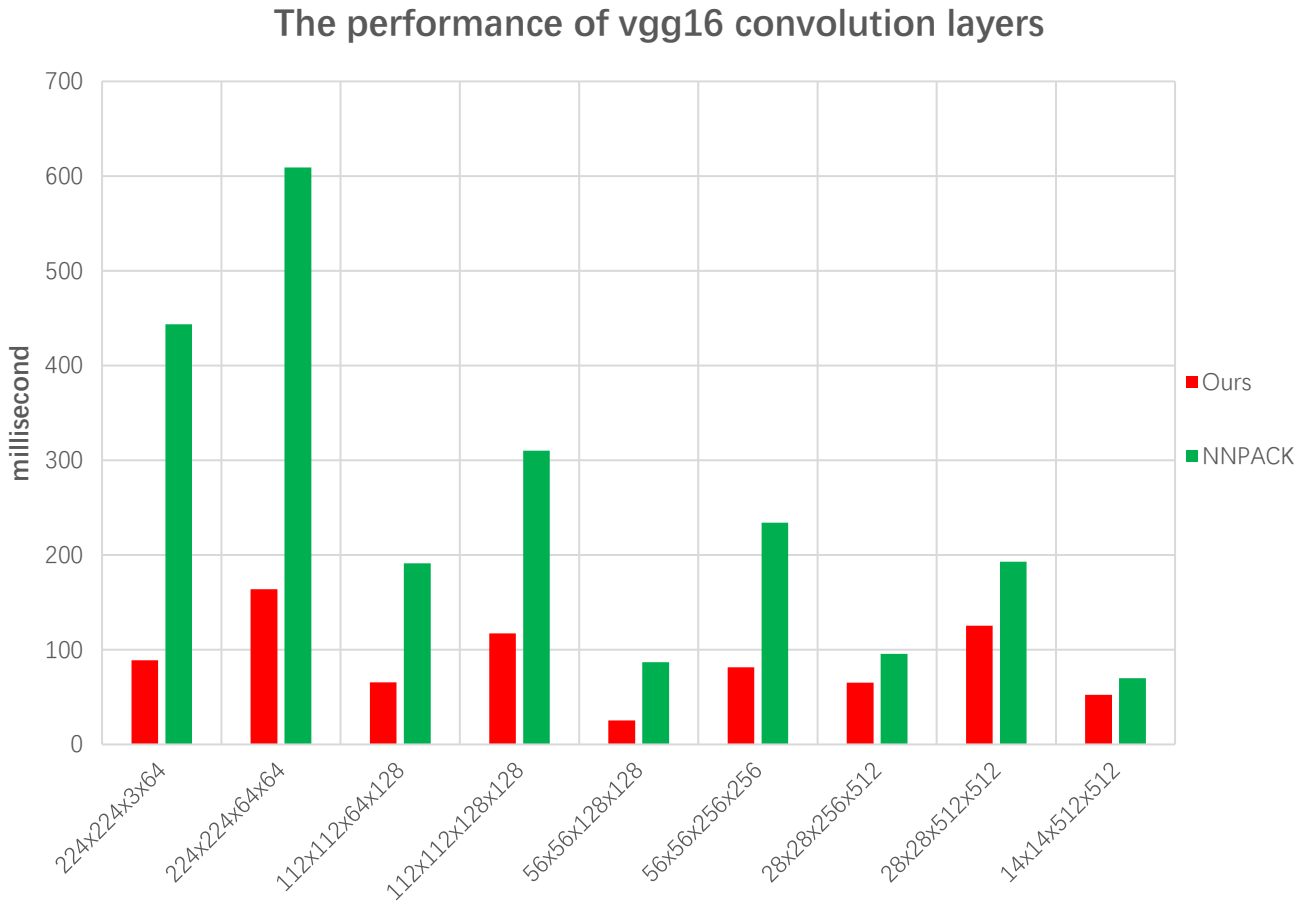
Single thread performance on ARM processor



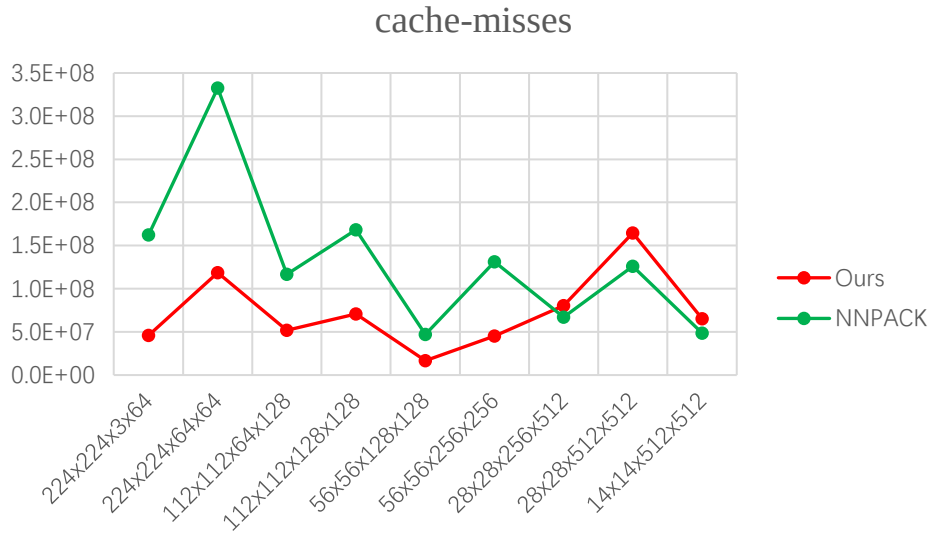
GEMM routine is not so efficient for some matrix shape, such as the last fewer layers.

*MKL-DNN does not support Winograd convolution on ARM processor.

Multi threads performance on ARM processor



The performance of VGG16 convolution layers



Batch size is 64, thread number is 32. We use $Perf\ stat - e\ cache - misses$ to perform profiling.

Conclusion

- Winograd-based algorithm performance is more easily restricted by transform operations
  - Reorder data layout to reduce non-sequential memory access
  - Reduce memory replication (using pointer movement to replace it)

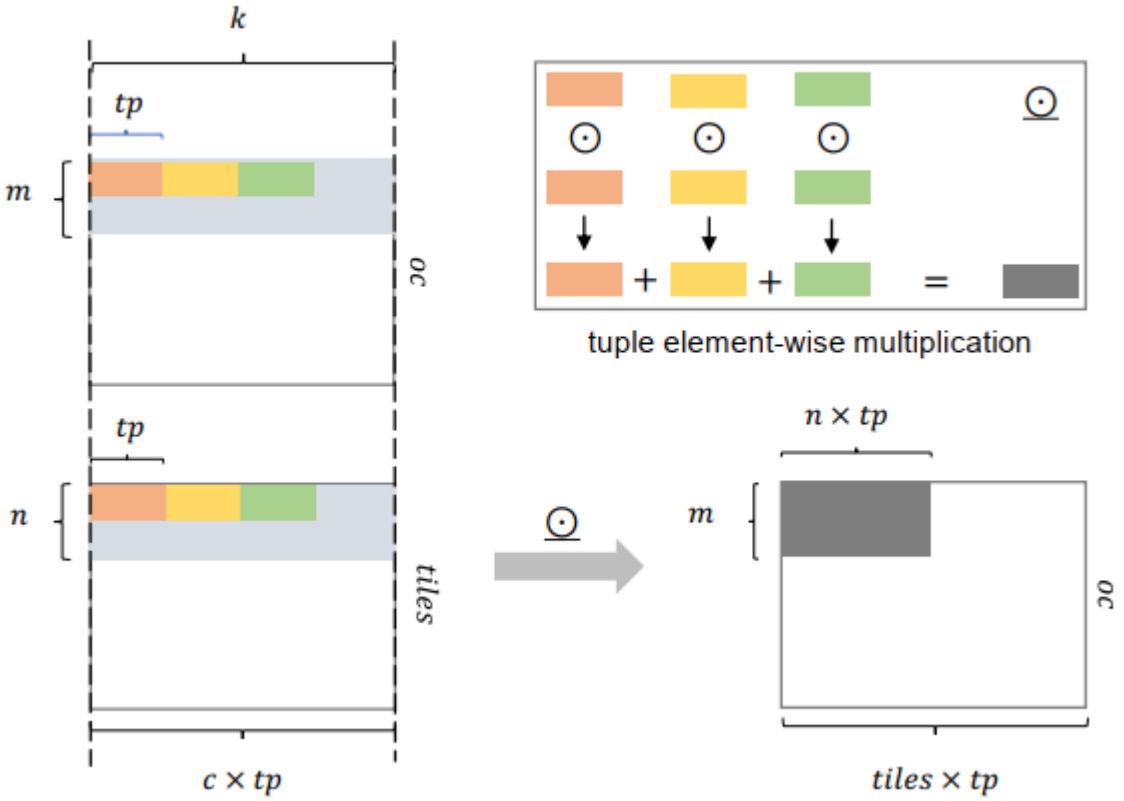
- ARM manycore CPU has more cores, NUMA nodes and the parallel performance is more easily restricted by memory bandwidth, cache contention and remote access latency
  - Use divide-and-conquer algorithm to construct independent tasks
  - Each thread reuses a boundary-aligned memory buffer
  - Create a run queue based on OpenMP, the queue can specify different types of data for NUMA scheduling

- The techniques used for GEMM optimization can be similarly applied to our method

Discussion

The arithmetic intensity of the algorithm:

$$b_1 = \frac{M}{m}, b_2 = \frac{N}{n}, AI = \frac{b_1 b_2 mn\left(\frac{k}{tp} + k\right)}{b_1 b_2 (mk + nk)} = \frac{mn(\frac{1}{tp} + 1)}{m + n}$$



It is not efficient for x86 AVX512 (long vector).
For example, $m = n = 64, tp = 4$,
AI = 64*64*(1/4+1)/(64+64)=40FLOPs/byte

When $tp = 16$, we need to reduce m and n to maintain block into L2 cache, $m = n = 16$,
AI = 16*16*(1/16+1)/(16+16)=8.5FLOPs/byte

Adopt $nchw8c$ or $nchw16c$ (MKL-DNN's data layout)?

Q&A

Thanks for your attention!

Questions ?