Deep Learning on Large-scale Multicore Clusters

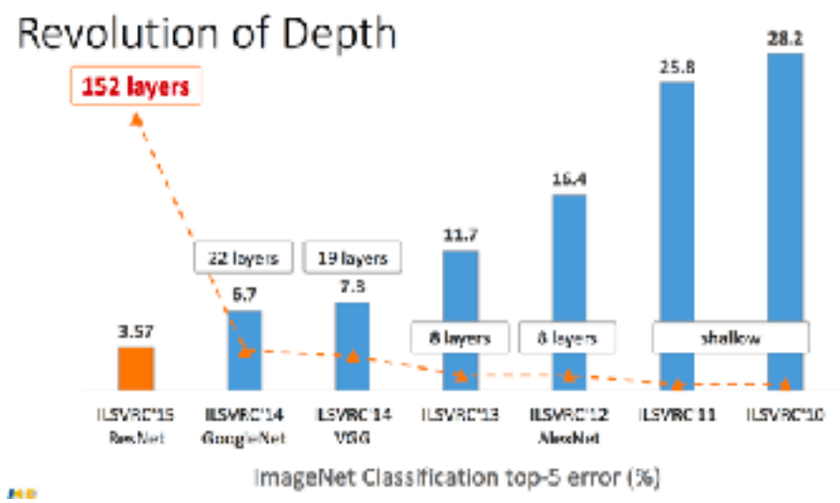
Kazumasa Sakiyama[1], Shinpei Kato[1], Yutaka Ishikawa[2], Atsushi Hori[2], Abraham Monrry[3]

[1]The University of Tokyo
[2]RIKEN
[3]Nagoya University

Background

- Application of Deep Learning to various areas
  - Image recognition, Voice recognition, Natural language processing, etc.

- Appearance of massive and deep neural networks
  - ResNet : 152 layers



*Kaiming He with Xiangyu Zhang, Shaoqing Ren, Jifeng Dai, & Jian Sun
Microsoft Research Asia (MSRA)

Background

CNN(Convolutional Neural Network)

Neural network which is specialized to pattern recognition

GPU is mainly used for accelerating CNNs. [1]

Most massively parallel systems are **CPU**-based.[2]

[1]Dominik Scherer, Hannes Schulz, and Sven Behnke. Accelerating Large-Scale Convolutional Neural Networks with P
Graphics Multiprocessors, pages 82–91. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
[2]K Computer http://www.aics.riken.jp/jp/k/

Problems

- Scalability of parallelized CNNs
  - Multi-core CPUs
  - Massively parallel systems

- Inefficient usage of resources at run time on CPUs

- Adaption of the methods of multi-GPUs[3]

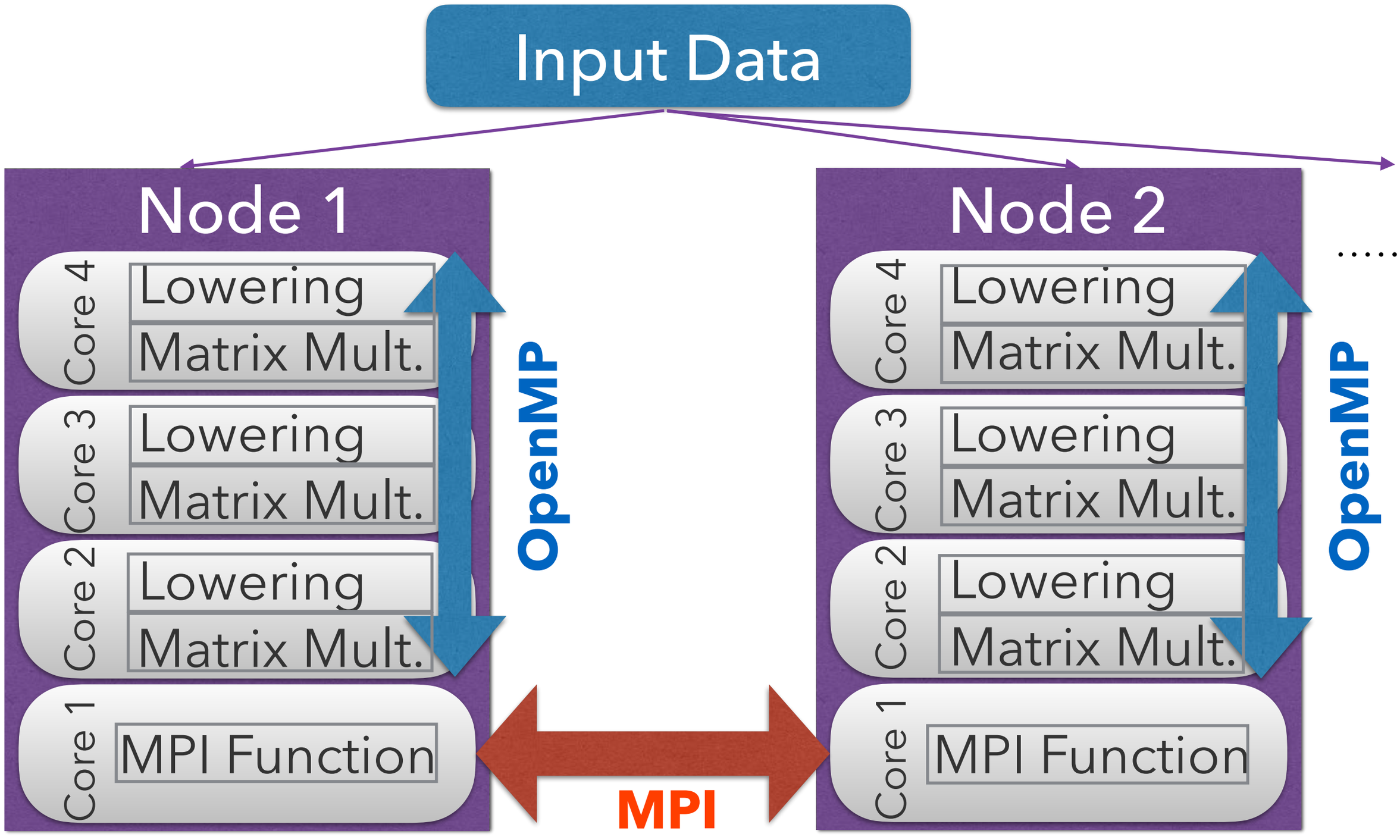
[3] Alex Krizhevsky. One weird trick for parallelizing convolutional neural net- works. CoRR, abs/1404.5997, 2014.

Various accelerating methods on GPUs

- More parallelism with simultaneous training on multiple data(batch)[4]
- Lowering Convolution[5]
- Parallelization across multi-GPUs[6]
  - Data parallelism: Distributed data, multiple networks
  - Model parallelism: Shared data, single network
  - Hybrid parallelism:Combination of them

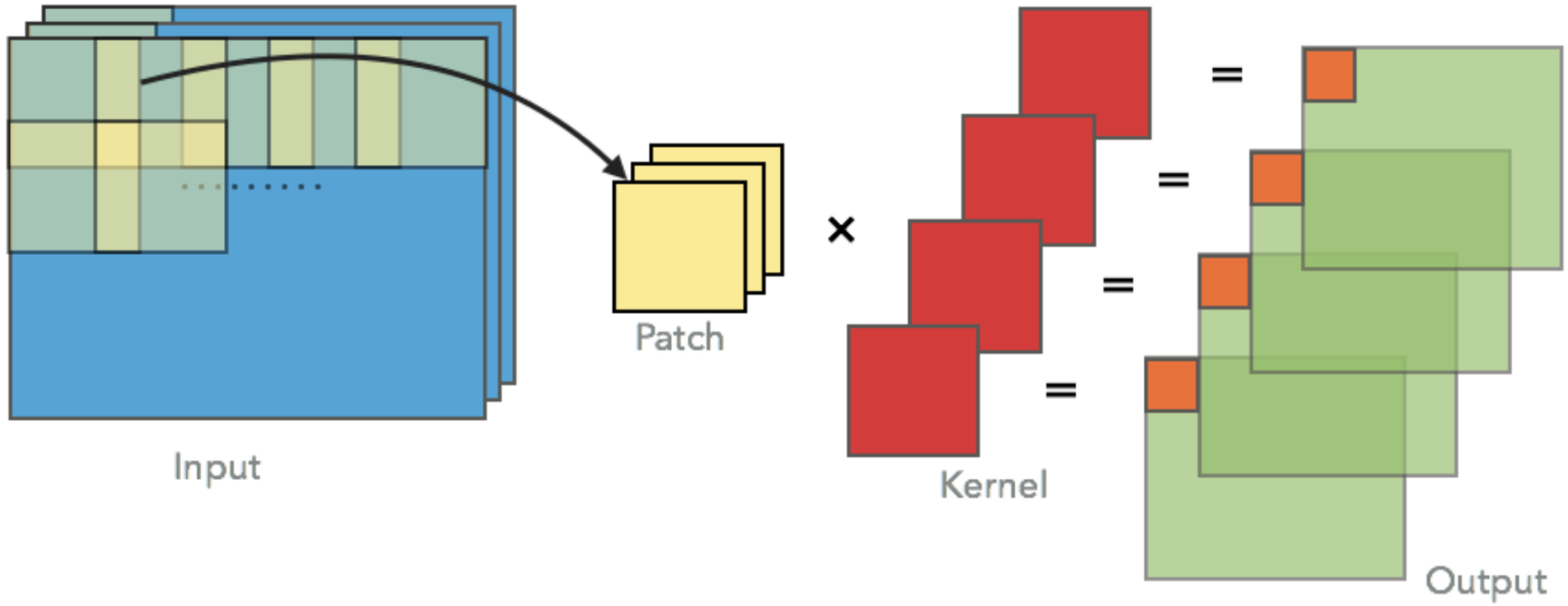
[4]D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. Neural Netw., 16(10):1429–1451, De- cember 2003.
[5]Kumar Chellapilla, Sidd Puri, and Patrice Simard. High Performance Con- volutional Neural Networks for Document Processing. In Guy Lorette, ed- itor, Tenth International Workshop on Frontiers in Handwriting Recogni- tion, La Baule (France), October 2006. Universite de Rennes 1, Suvisoft. http://www.suvisoft.com.
[6] Alex Krizhevsky. One weird trick for parallelizing convolutional neural net- works. CoRR, abs/1404.5997, 2014.

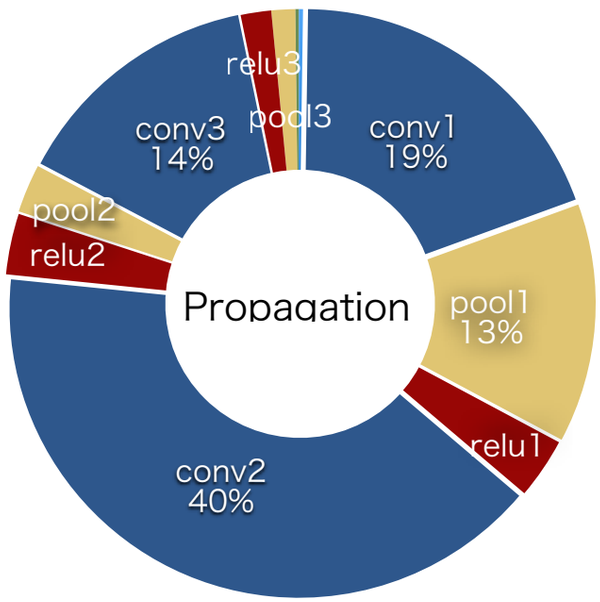
Implementation

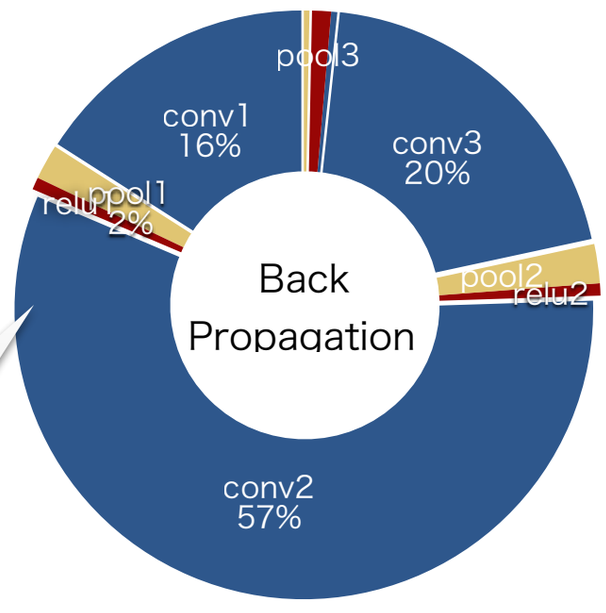


Implemented massively parallel CNNs and evaluated its scalability.

Convolutional layers







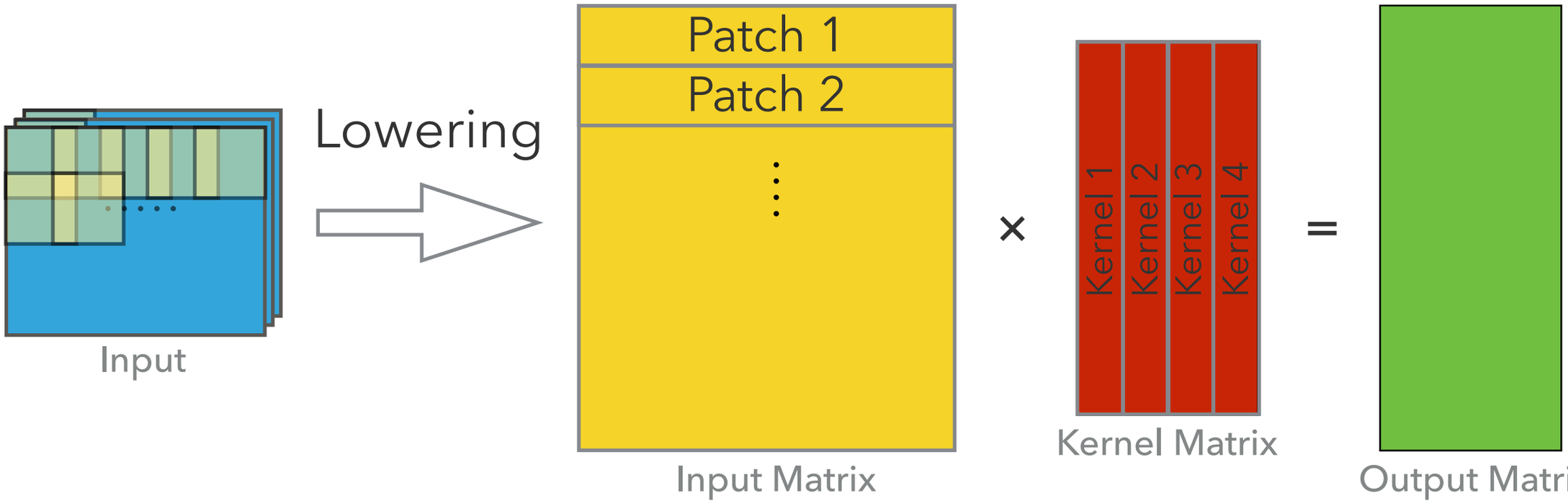
Considerably high computing costs

84% processing time of our model

Environment

- Xeon E5-2687 (3.4 GHz, 8 cores)
- CentOS release 6.4

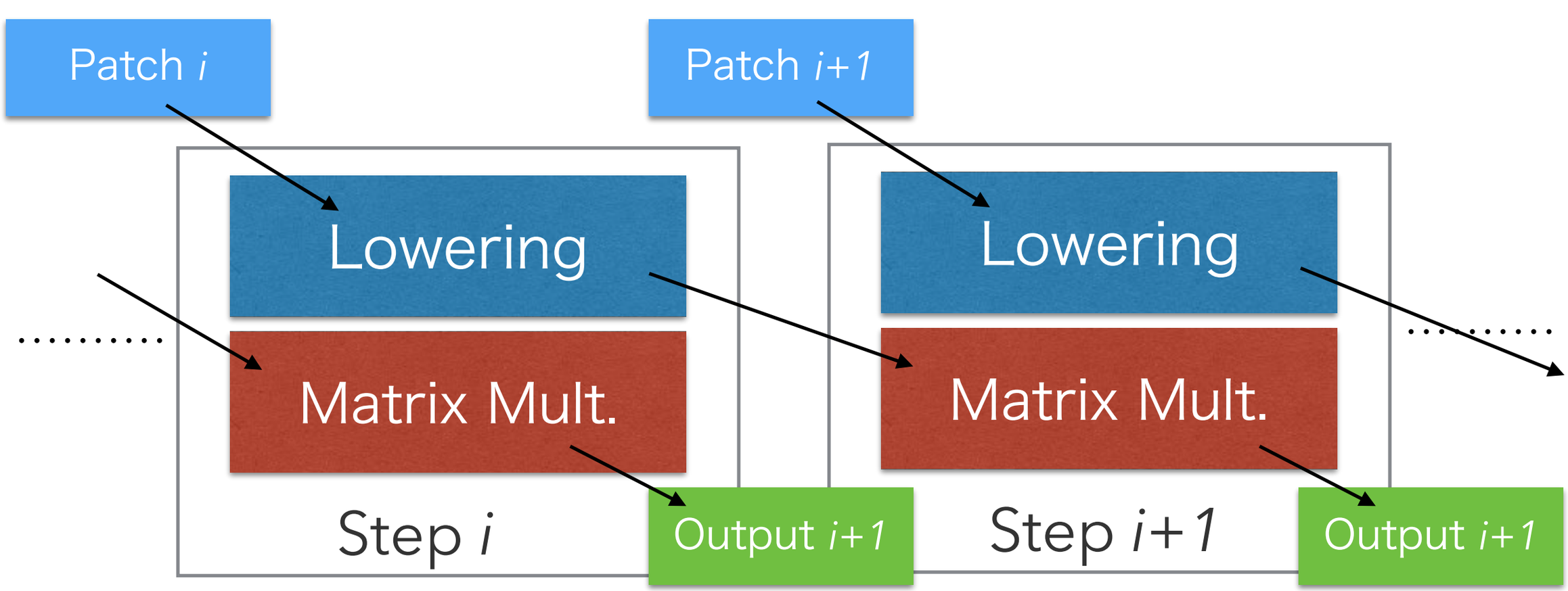
Lowering Convolution



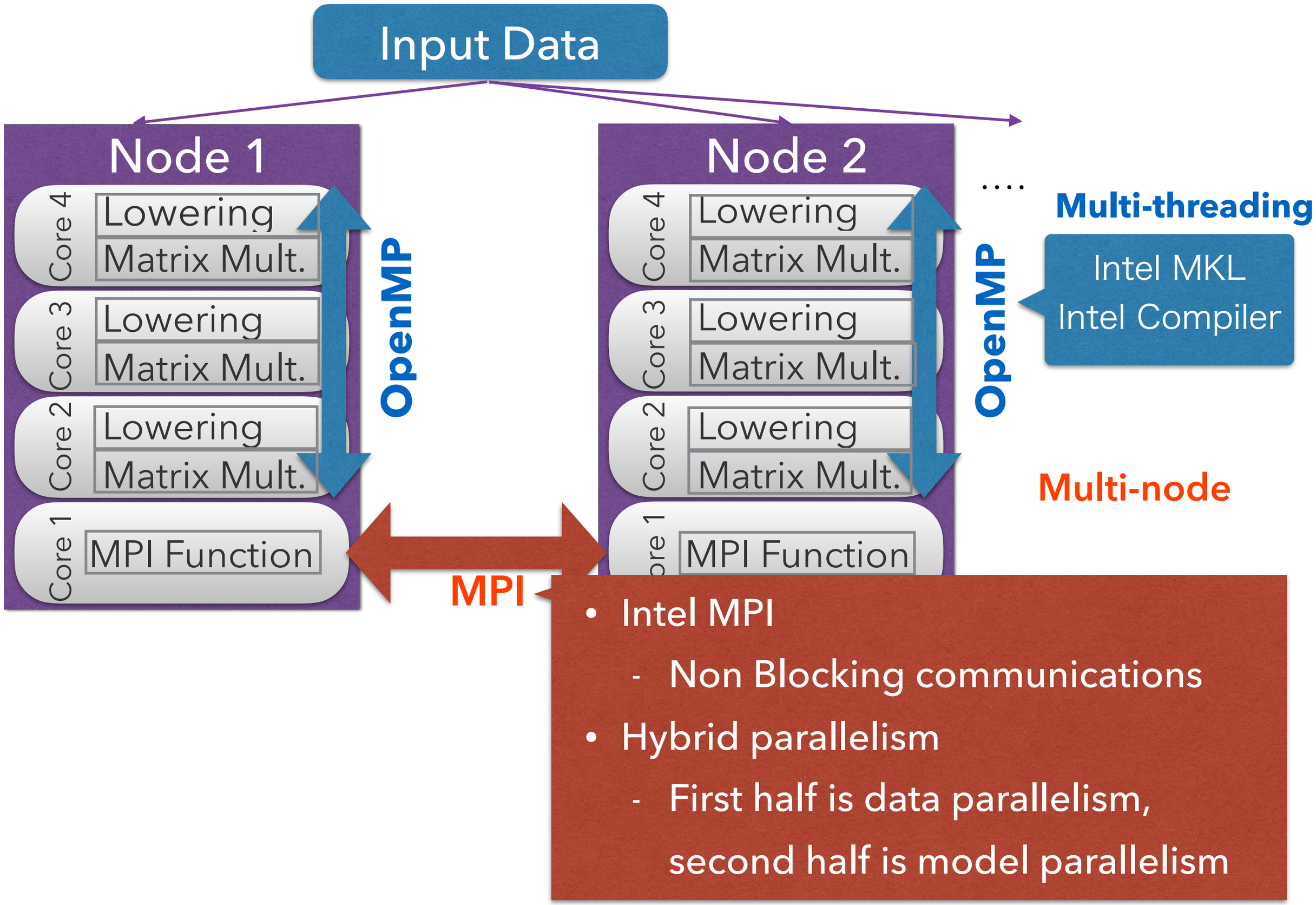
- Accelerating method of convolutions
- Stable performance with numerical calculation libraries
- Different bottlenecks
  - Lowerings : Memory access
  - Matrix multiplications : Floating-point operations

Pipelined Lowering

- Subdividing processes
- Simultaneous execution of Lowering and matrix multiplication in one core
- Efficient usage of resources



Implementation



Experiment of multi-thread parallelizations

Comparison of three methods for convolutional layers

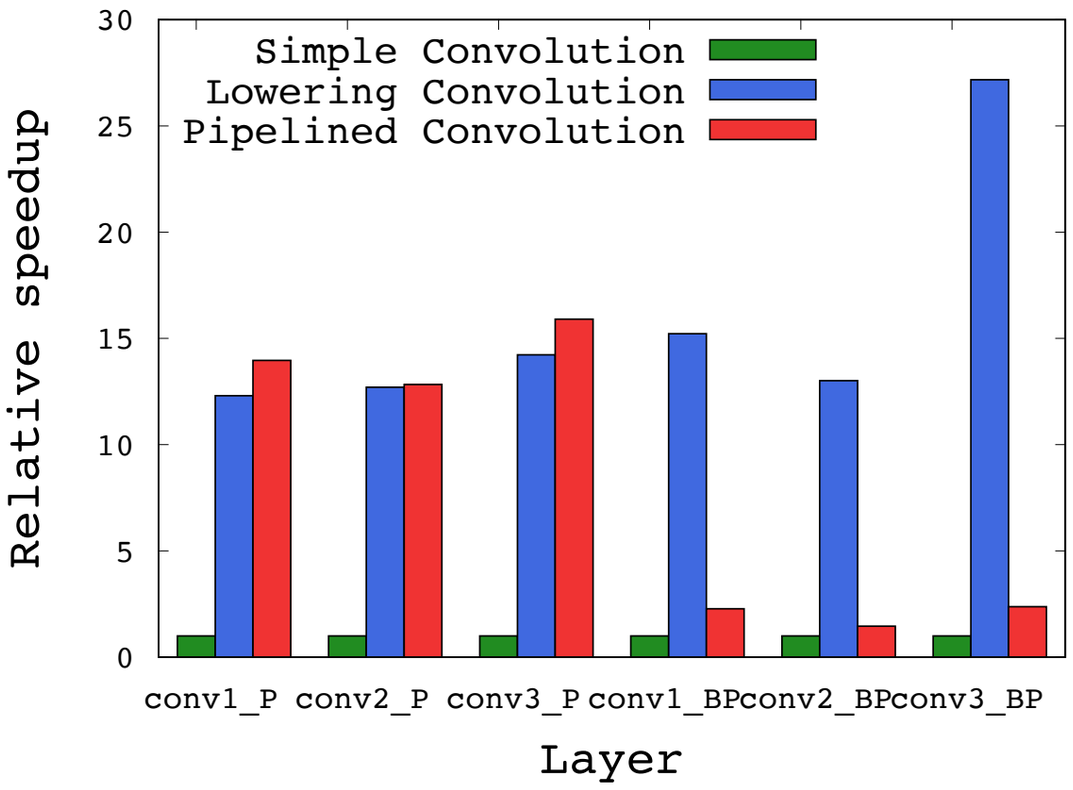
- Simple convolution
- Lowering convolution
- Pipelined lowering convolution

Environment

- Xeon E5-2687 (3.4 GHz, 8 cores) x2
- NUMA architecture
- CentOS release 6.4

Evaluation on one 8-core CPU

- Stable and high performance with lowering convolution
- Significant performance decrement with:
  - Simple convolution
  - Pipelined convolution applied to backpropagations

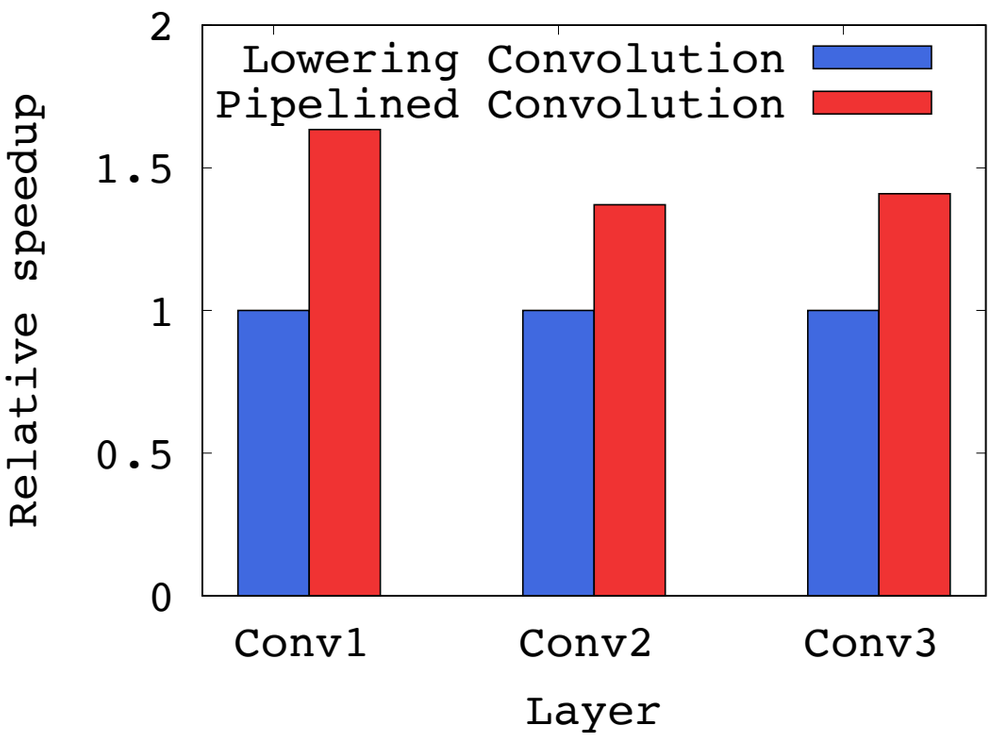


- 3 convolutional layers in CIFAR-10
- Batch size: 100
- *_P is propagation of convolution
- *_BP is backpropagation of convolution

Evaluation on two 8-core CPUs

- Plus factors
  - High locality of data, less communications across memories
  - Effective use of HTT (Hyper-Threading Technology)

- Minus factors
  - Fractionation of processes
  - The difference in processing time between lowering and matrix multiplication



- 3 convolutional layers in CIFAR-10
- Batch size: 100
- NUMA architecture
- At most 1.64x speedups

Evaluation with a deep and complicated network

- Evaluation on all convolutional layers in GoogLeNet
- Significant performance decrement on some layers which has large weight size relative to sizes of lowered inputs

Kernel size	Input size	Lowered size	Weight size	Output size	Speedup
7x7	150,528px	1,843,968px	9,408px	802,816px	1.27x
3x3	200,704px	1,806,336px	110,592px	602,112px	1.16x
3x3	75,264px	677,376px	110,592px	100,352px	0.93x
5x5	12,544px	313,600px	12,800px	25,088px	1.23x
3x3	100,352px	903,168px	221,184px	150,528px	0.85x
5x5	25,088px	627,200px	76,800px	75,264px	0.96x
3x3	18,816px	169,344px	179,712px	40,768px	0.66x
5x5	3,136px	78,400px	19,200px	9,408px	2.04x
3x3	21,952px	197,568px	225,792px	43,904px	0.65x
5x5	4,704px	117,600px	38,400px	12,544px	1.15x
3x3	25,088px	225,792px	294,912px	50,176px	0.60x
5x5	4,704px	117,600px	38,400px	12,544px	1.15x
3x3	28,224px	254,016px	373,248px	56,448px	0.56x
5x5	6,272px	156,800px	51,200px	12,544px	1.14x
3x3	31,360px	282,240px	460,800px	62,720px	0.54x
5x5	6,272px	156,800px	102,400px	25,088px	0.81x
3x3	7,840px	70,560px	460,800px	15,680px	0.03x
5x5	1,568px	39,200px	102,400px	6,272px	0.04x
3x3	9,408px	84,672px	663,552px	18,816px	0.03x
5x5	2,352px	58,800px	153,600px	6,272px	0.04x

Experiment of multi-node parallelizations

Evaluating scalabilities of basic multi-node parallelization techniques for CNNs

- Data parallelism
- Hybrid parallelism (a combination of data and model parallelism) by Krizhevsky, et al.[3]

Environment A

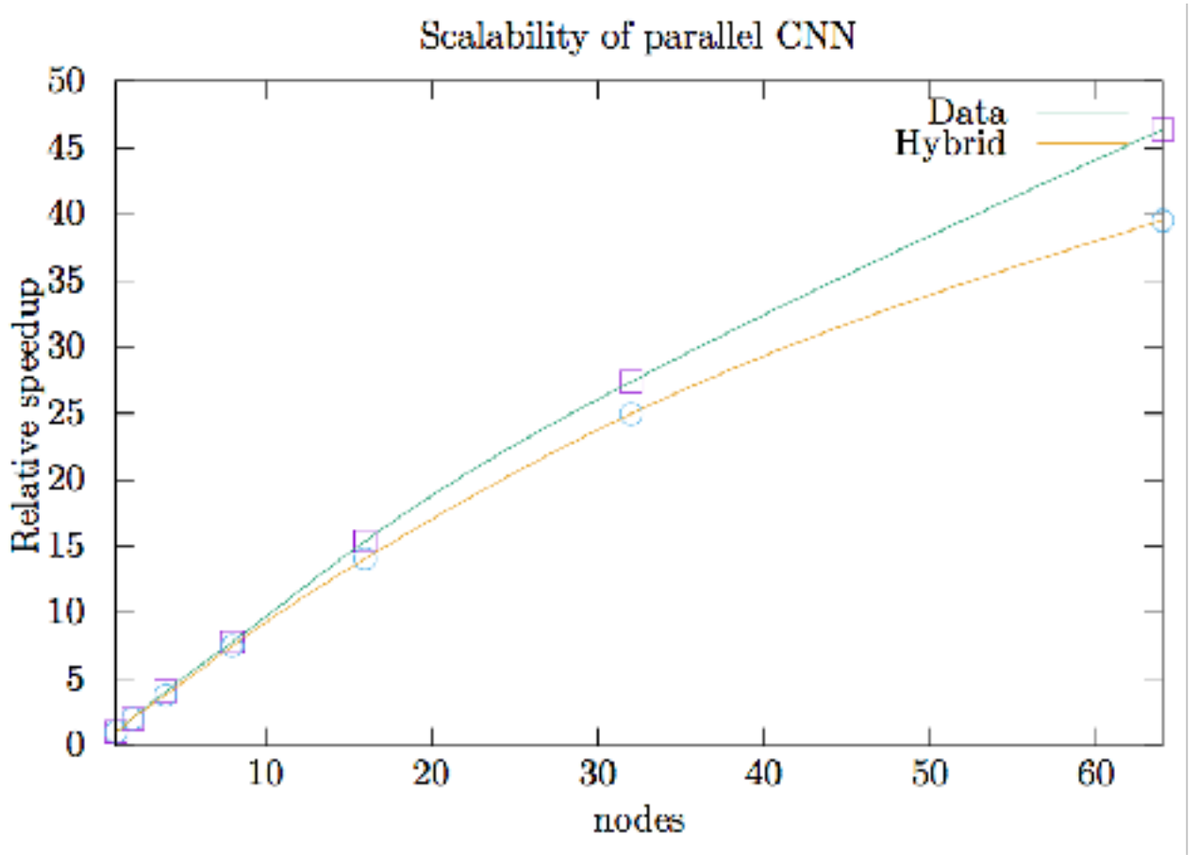
- Xeon E5 2600 v3 (2.6 GHz, 14 cores, 2 CPU/node)
- Red Hat Enterprise Linux 6.5
- 64 nodes

Environment B

- Fujitsu SPARC64 Xlfx processors (2.2 GHz, 16 cores, 2 CPU/node)
- Designated OS for Fujitsu architecture
- 864 nodes

Evaluation on environment A (small cluster)

- Data parallelism showed higher scalability
- Data parallelization linearly scales when communication time can be overlapped by execution time
- Hybrid parallelism is considered to perform well when communication time becomes bottleneck

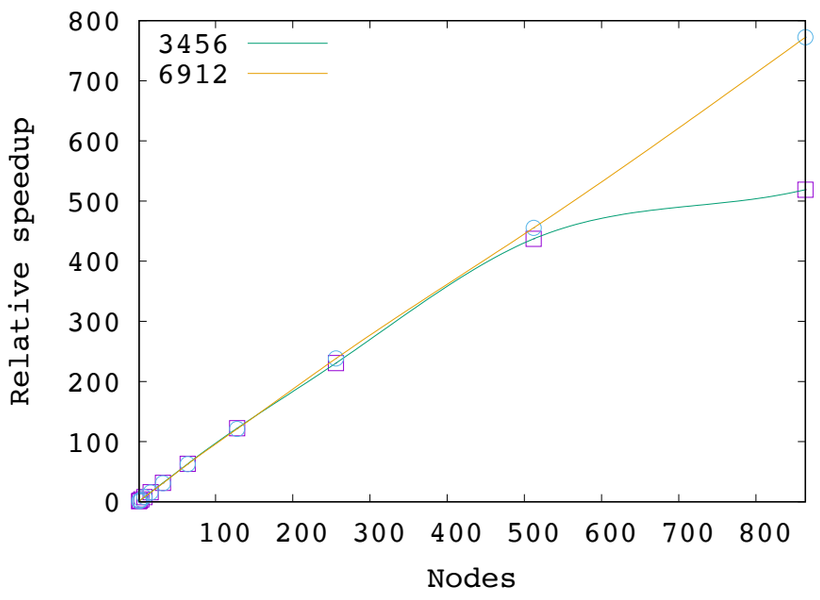


- Batch size: 256
- Image size: 3 x 120 x 160

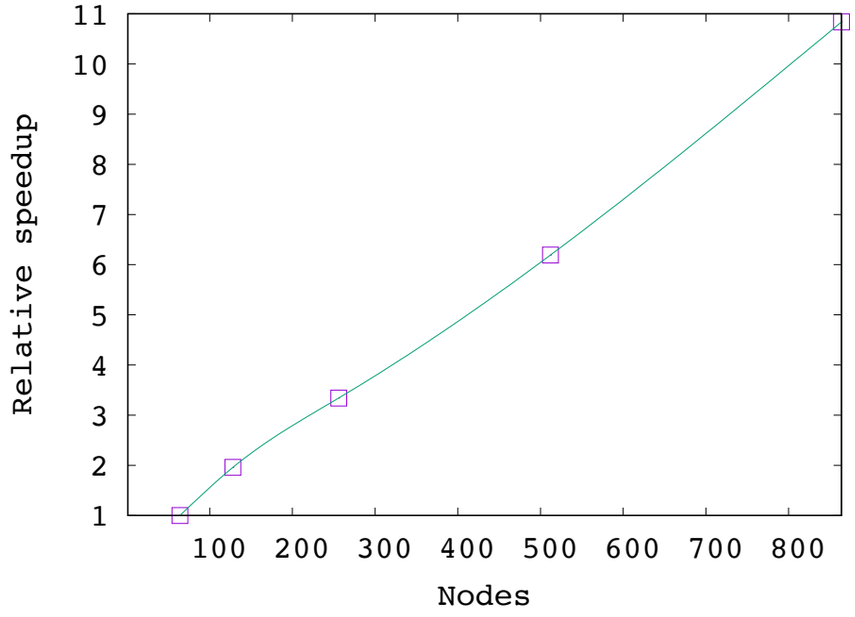
Evaluation on environment B (large cluster)

- Scalability of data parallelism depends on batch size
- Valid batch size is at most 8192 [7]
- Performance linearly scales with
  - a batch size of 6912 when input size is small (left figure)
  - a batch size of 864 when input size is large (right figure)

Scalability with image size of 3x120x160



Scalability with image size of 3x1920x1080, batch size of 864



[7] Priya Goyal, Piotr Dolla´r, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017.

Conclusion

- Pipelined convolution achieved at most 1.64x speedups relative to conventional convolution

- Performance of pipelined convolution decrements with a layer which has a large size of weight

- When CNNs are parallelized on a multi-core cluster, performance scales linearly with data parallelization in most cases

Future work

- Parallelization over 1000 nodes
- Evaluation that takes learning efficiency into account
- Evaluation with various network models
- Optimizations for small batch size per node

Thank you