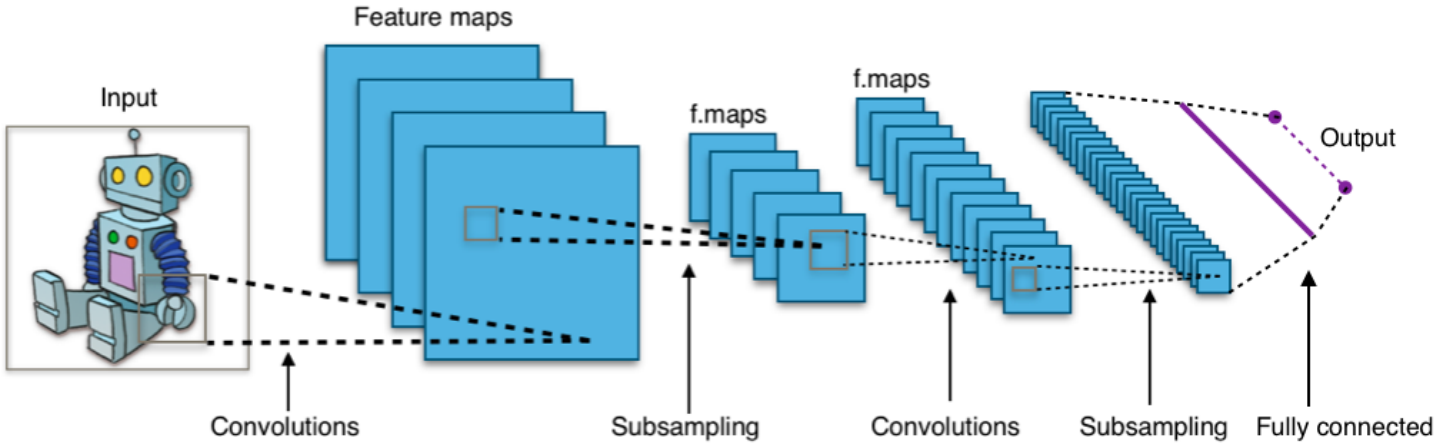
A Network-Centric Hardware/Algorithm Co-Design to Accelerate Distributed Training of Deep Neural Networks

Youjie Li[1], **Jongse Park**[2], Mohammad Alian[1], Yifan Yuan[1],
Zheng Qu[3], Peitian Pan[4], Ren Wang[5], Alexander Schwing[1],
Hadi Esmaeilzadeh[6], Nam Sung Kim[1]


[1]UIUC, [2]Georgia Tech, [3]THU,
[4]SJTU, [5]Intel, [6]UCSD

Deep learning is growing exponentially!

Training time (weeks or months)

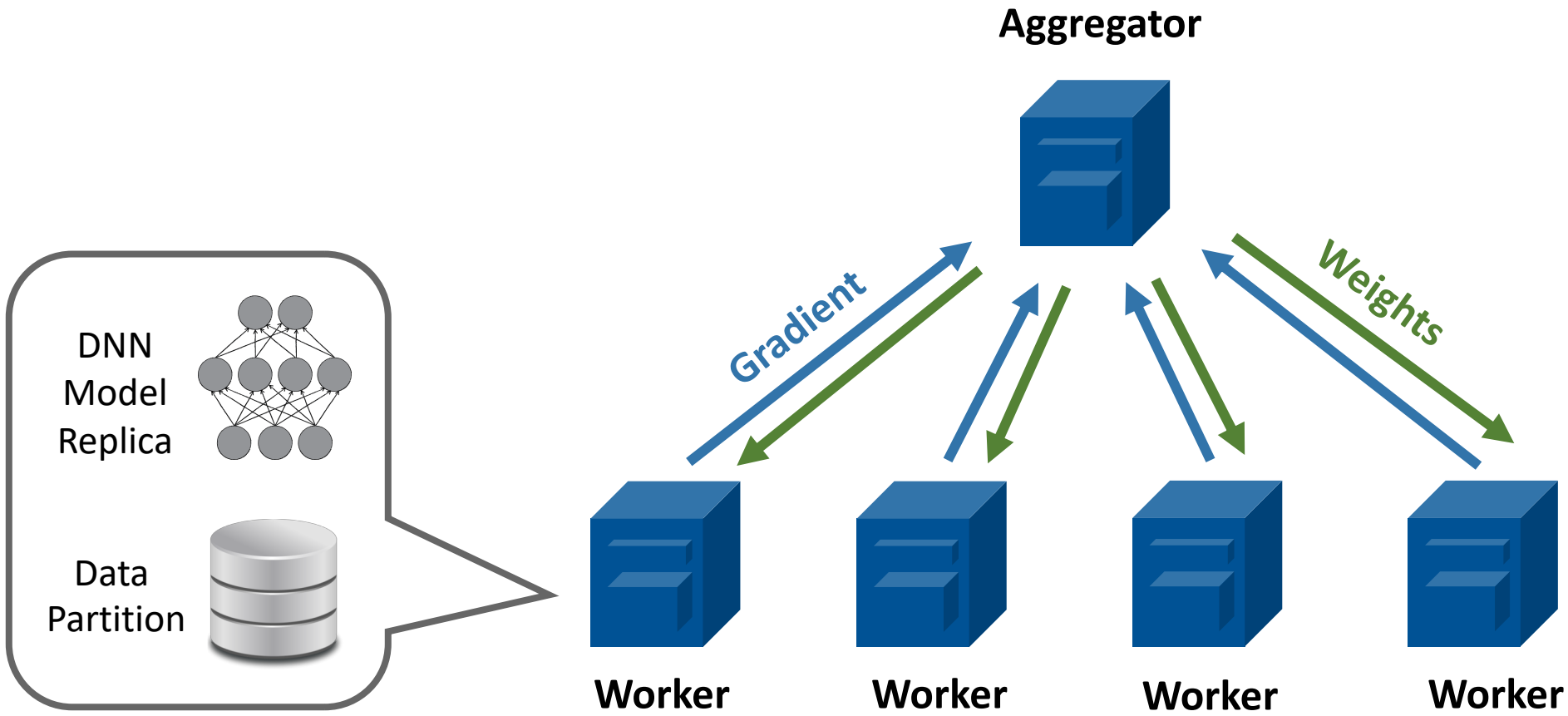


Training data size (hundreds of TB)

DNN model complexity (hundreds of MB)

Distributed learning is essential!

Parallelizing the learning task over multiple nodes.



TensorFlow

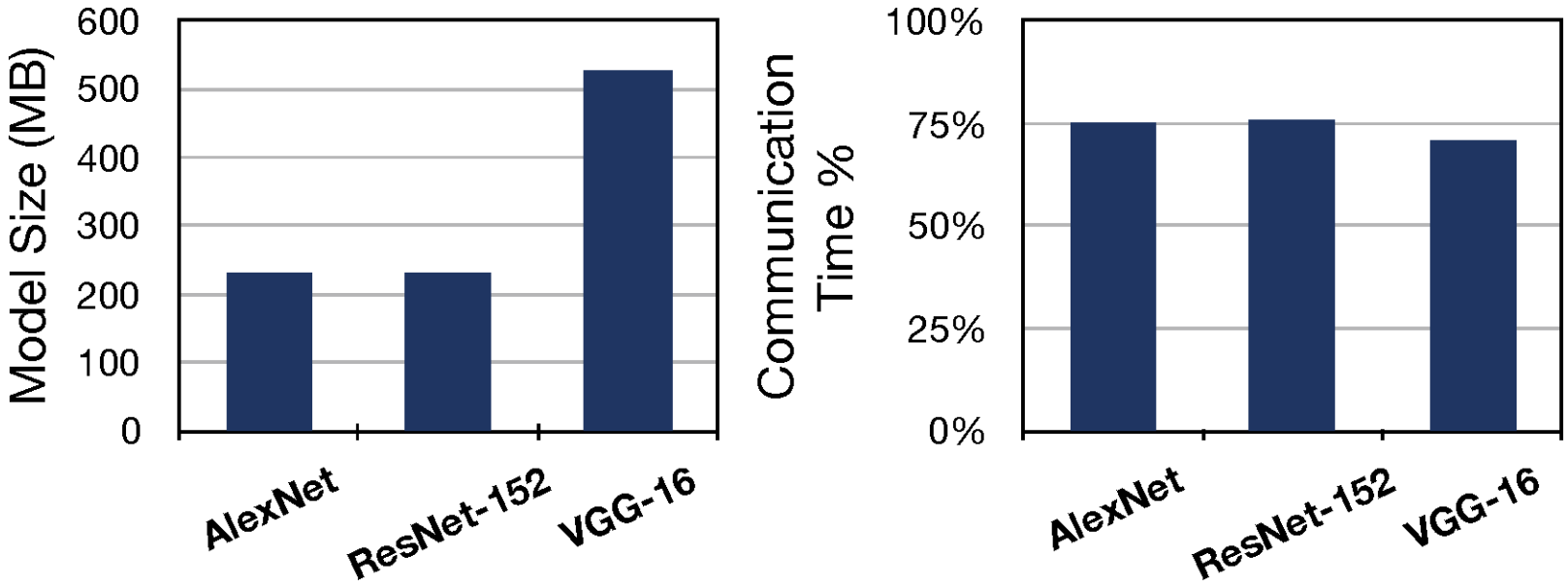
Caffe2

mxnet

PaddlePaddle

DL4J

Significant communication overhead in distributed learning!

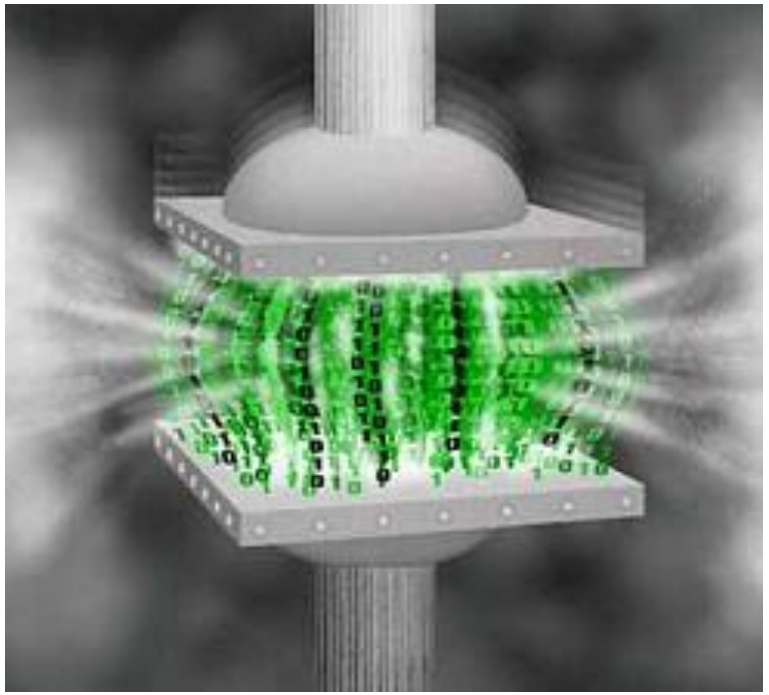


INCEPTIONN framework

- Synchronous training equivalent to TensorFlow
- Five Nodes
- 10 Gb Ethernet
- OpenMPI 2.0
- Titan Xp GPUs
- CUDA 8.0

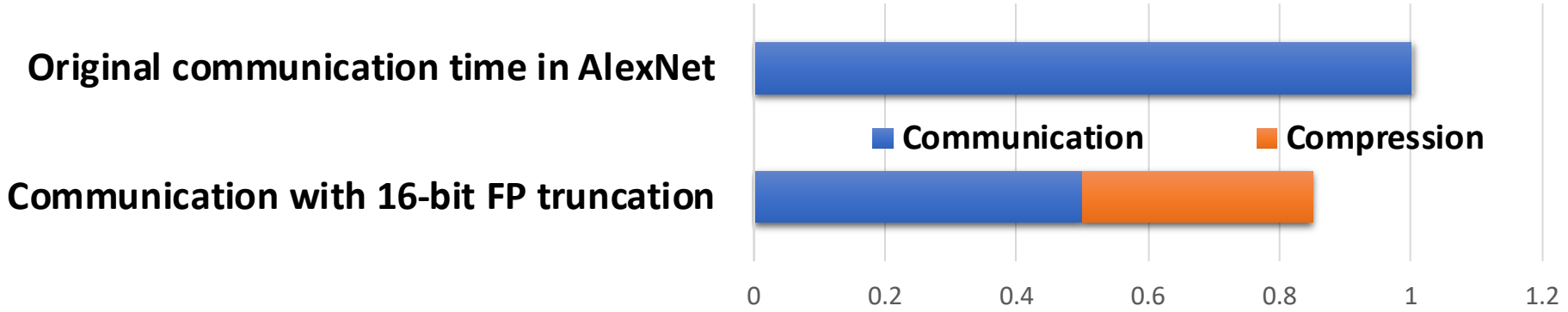
How to reduce communication?

Straightforward Solution: Compression



Challenges for compression

- **Challenge #1: Expensive compression overhead**



- **Challenge #2: Limited compressibility of weights**

Train AlexNet with 16-bit FP truncation	Accuracy
Baseline without truncation	80.2 %
Weight truncation Only	00.9%
Gradient truncation Only	79.7%

INCEPTIONN

A hardware/algorithm co-design to accelerate distributed training

Gradient-centric, decentralized training algorithm

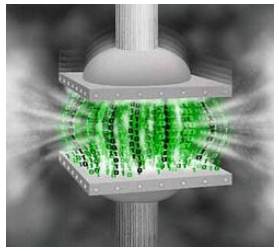
Hardware-friendly lossy gradient compression algorithm

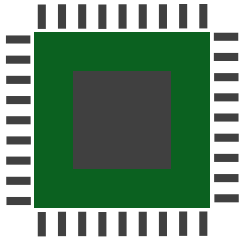
In-network accelerator for compression

In-network acceleration

Pushing the compression to network

Conventional practice:

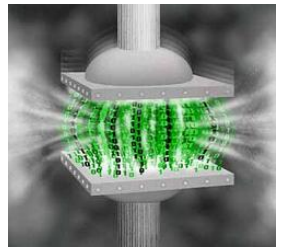




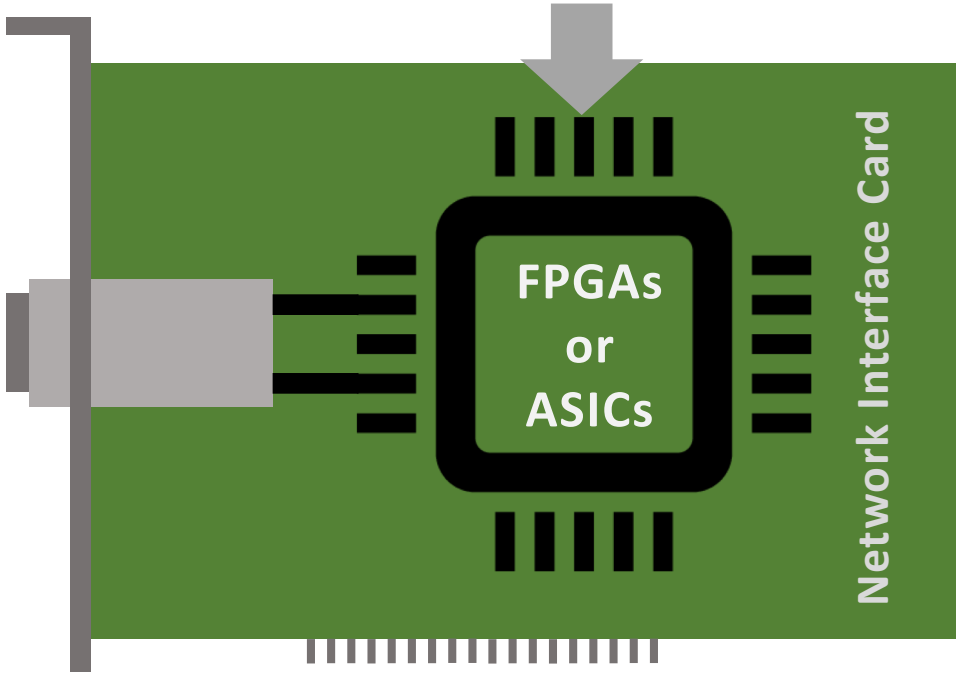
General-purpose processors

Heavy overhead

INCEPTIONN:



Light overhead



Hardware-friendly lossy gradient compression

Requirements:

- **High compression ratio**
- **Hardware-friendliness for acceleration**
- **Minimal loss in training accuracy**

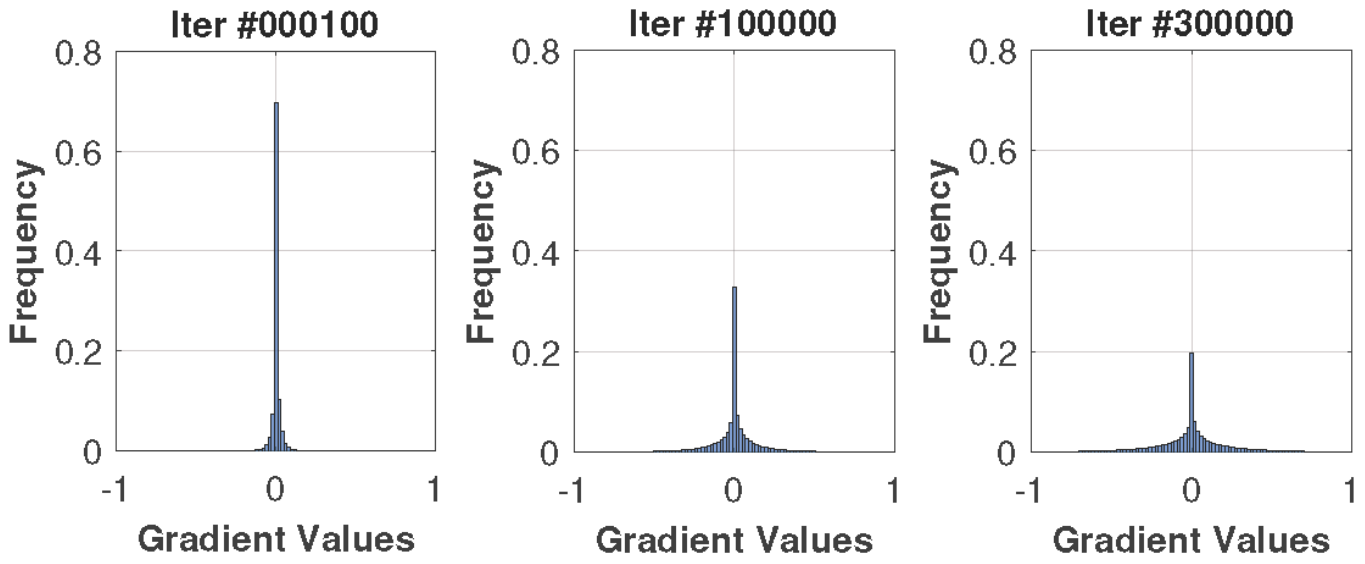
Solution:

- **Customized lossy compression algorithm for *gradients***

Why gradients?

High error resilience

Limited range and skewness to zero



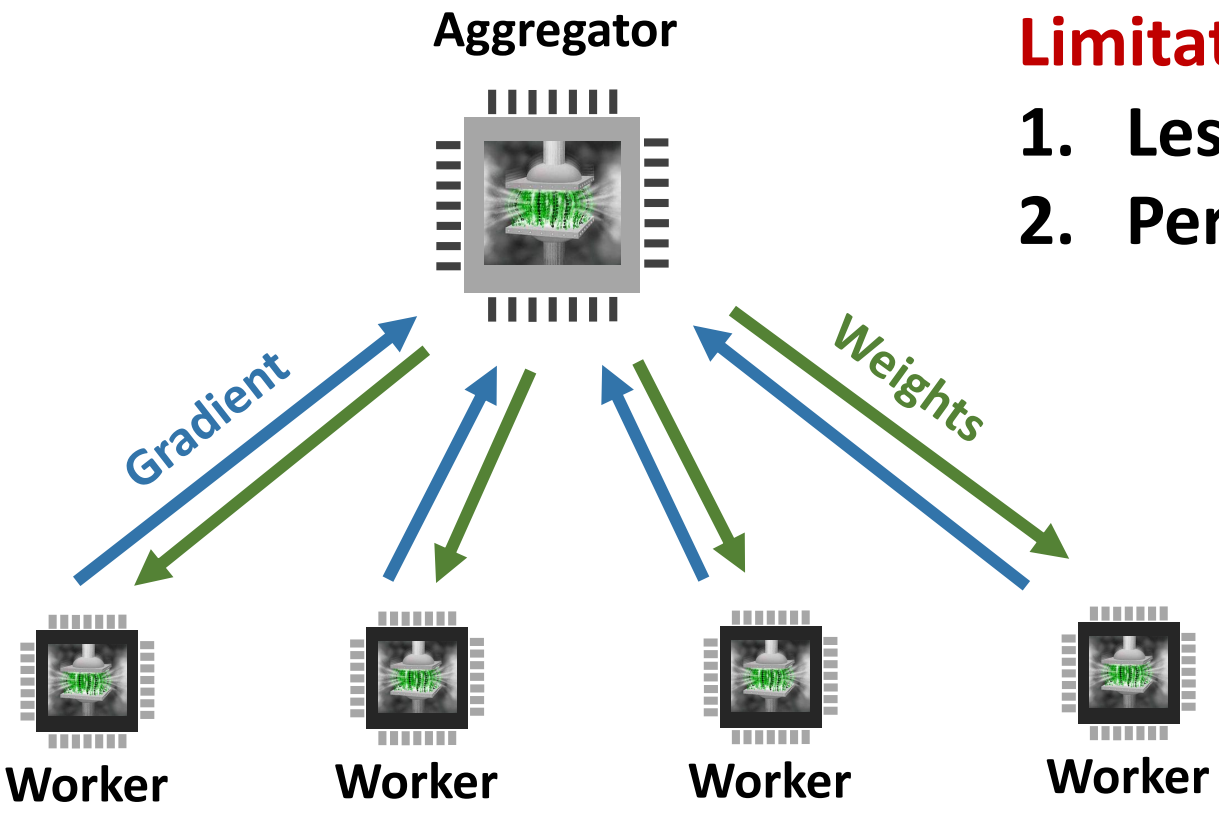
Early **Middle** **Late**

AlexNet gradient distribution

Key ideas for lossy gradient compression algorithm

- Remove exponents in FP representation by setting it to a constant
- Remember the diff by shifting on mantissa with a concat’ed 1
- Compress more aggressively as values are close to zeros

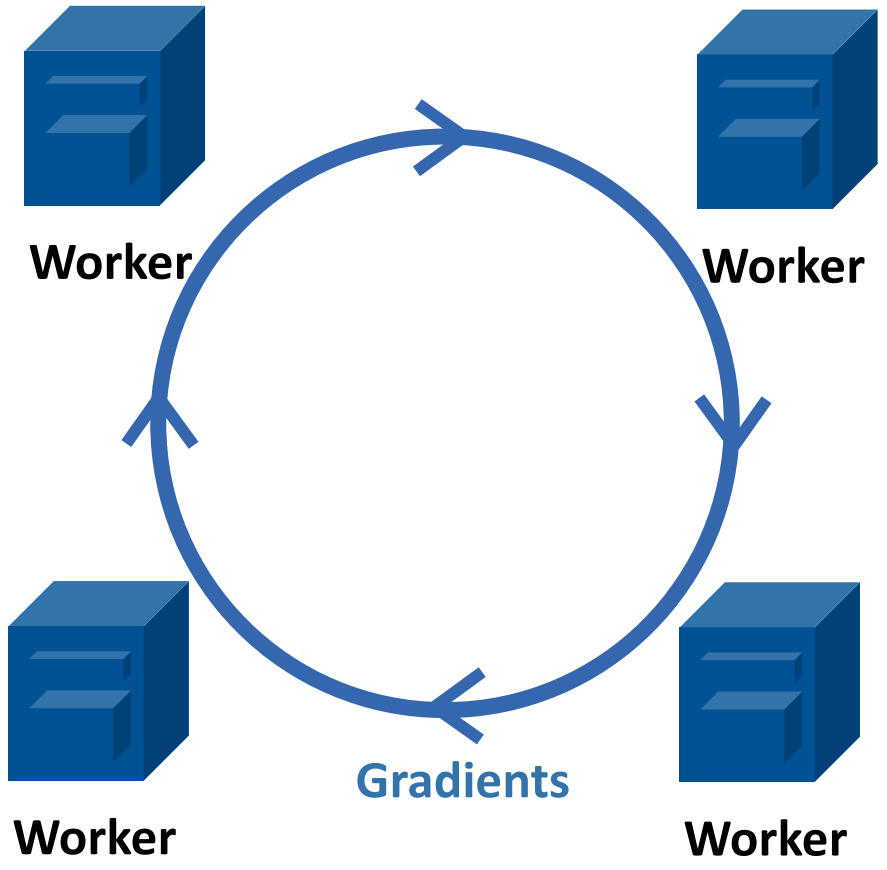
Compression with worker-aggregator approach



Limitations

1. Less opportunities for compression
2. Performance bottleneck at aggregators

Gradient-centric decentralized training



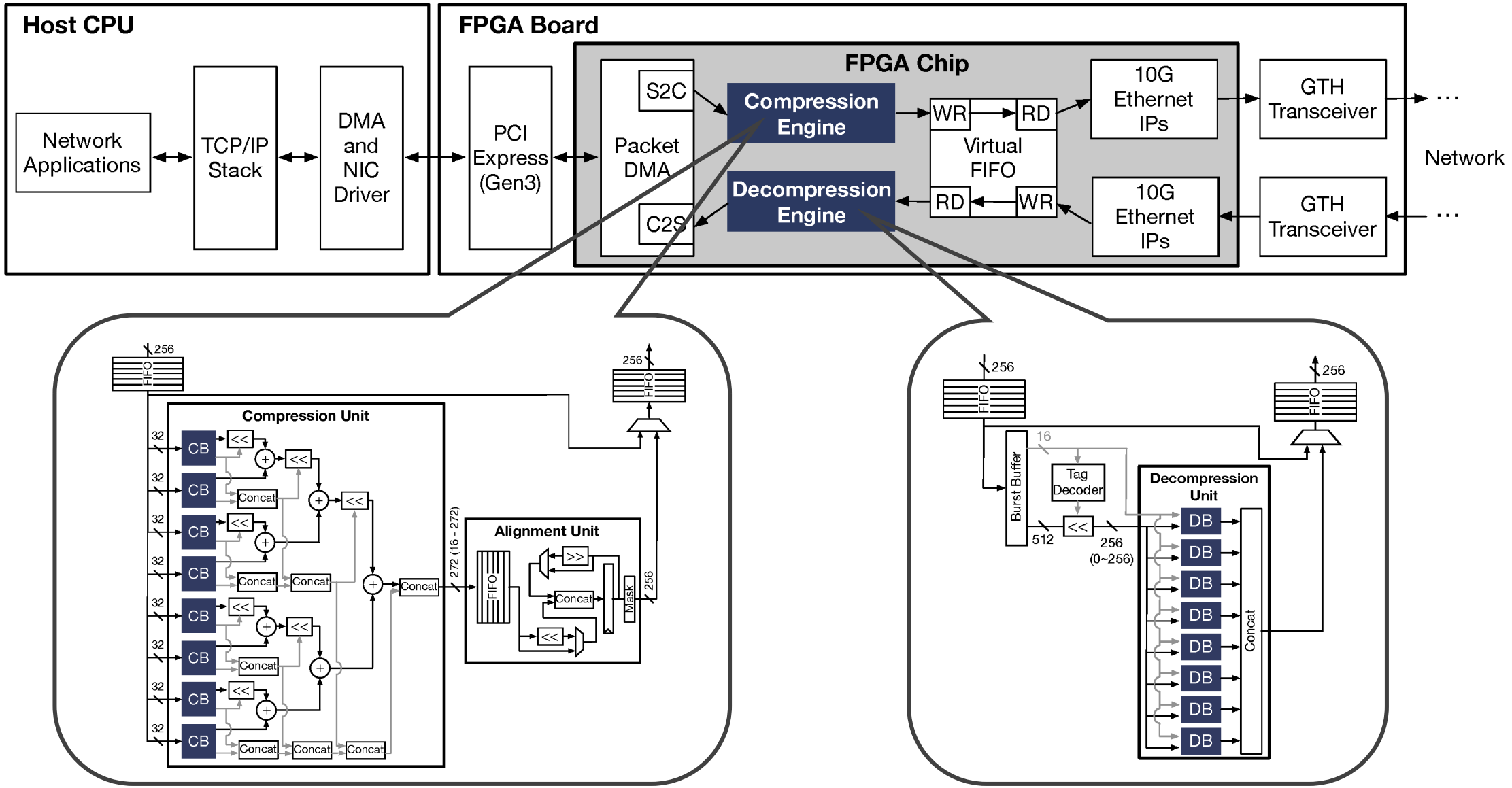
Approach

1. Communicate only gradients
2. Evenly distribute aggregation to the workers

Advantages

1. Maximize opportunities for compression
2. Balanced load for compression/decompression

Implementation

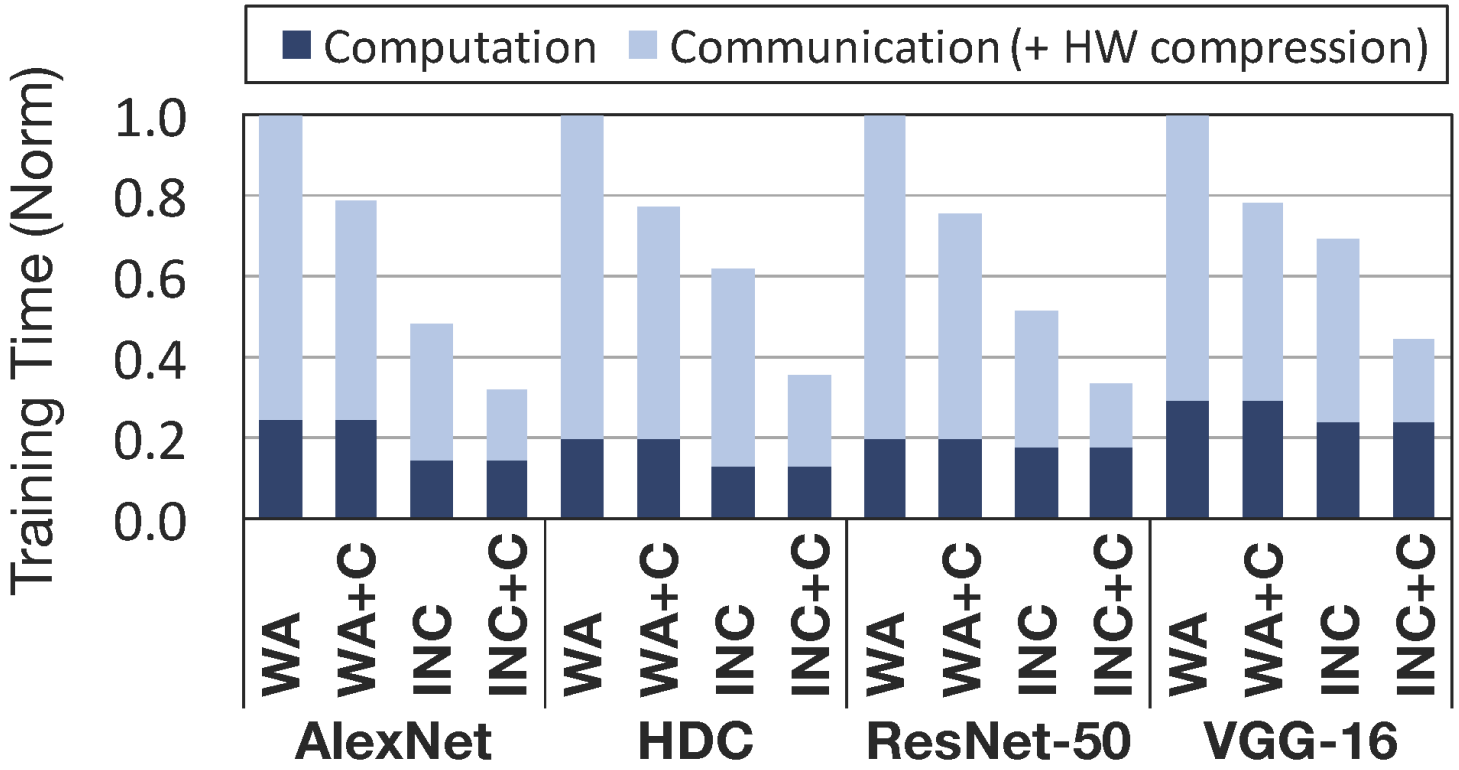


Evaluated DNN models and system specifications

Name	HDC	AlexNet	ResNet-50	VGG-16
Model size	5 MB	230 MB	100 MB	525 MB
Dataset	MNIST	ImageNet		
Mini-batch size	25	64	32	64
Training data size	60,000	1,281,167		

System specifications	
Number of nodes	4
System software	C++, CUDA 8.0, Intel MKL 2018, and OpenMPI 2.0.2
CPU	Intel Xeon CPU E5-2640 @2.6 GHz
GPU	NVIDIA Titan Xp
FPGA	Xilinx VC709 board
Network	10 Gb Ethernet

Training runtime comparison

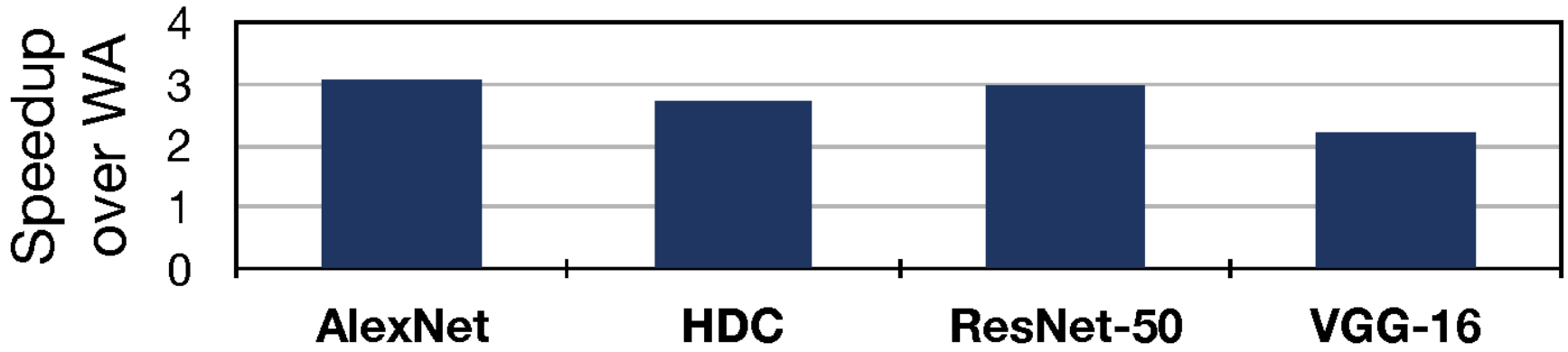


- WA: Worker-aggregator
- INC: INCEPTIONN
- WA+C: WA with compression
- INC+C: INC with compression

INC+C offers **76% lower communication time** compared to the WA baseline

INC+C offers **2.2~3.1x** system-level speedup over the WA baseline

Impact on final training accuracy



	AlexNet		HDC		ResNet-50		VGG-16	
	WA	INC+C	WA	INC+C	WA	INC+C	WA	INC+C
Training Time	175h	56h	170s	64s	378h	127h	847h	384h
# of Epochs	64	65	17	18	90	92	74	75
Final Accuracy	57.2%	57.2%	98.5%	98.5%	75.3%	75.3%	71.5%	71.5%

Only 1 or 2 more epochs are required to match the same level of accuracy

Conclusion

INCEPTIONN

Hardware-algorithm co-designed in-network acceleration solution to reduce the communication overhead in distributed DNN training