IEEE / ACM CCGRID 2019

2nd High Performance Machine Learning Workshop - HPML

Theoretical Scalability Analysis of Distributed Deep Convolutional Neural Networks

Adrián Castelló, Manuel F. Dolz, Enrique S. Quintana-Ortí, José Duato

UNIVERSITAT JAUME I

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

May 14th, 2019

Why Deep Learning now?

- Explosion in the amount of data available today
  - *Reliable training of the DNNs*
- Increased computing capabilities of current hardware
  - *Enables training in reasonable time*
- Significant algorithmic advances +
  development of open source frameworks for DNNs
  - *Facilitate the research and use of DNNs*
  - *Broaden the domains to which DNNs are being applied*

Deep Learning Applications

Key (and growing number of) applications:

- Text recognition and language translation,
- Image classification,
- Adaptive user profile,
- Voice recognition systems,
- Autonomous driving,
- Weather forecast, etc.

→ In general, social networks and big data analytics

Distributed training on HPC clusters

- Inference can be performed on low-end devices, but…
- training requires advanced HPC solutions
- So, what is important to achieve good performance?
  - Processor performance
  - Memory bandwidth
  - Network interconnect bandwidth
  - Number of cluster nodes
  - Parallelism model
  - Algorithm parameters

Modeling the performance
of CNNs Distributed Training

Index

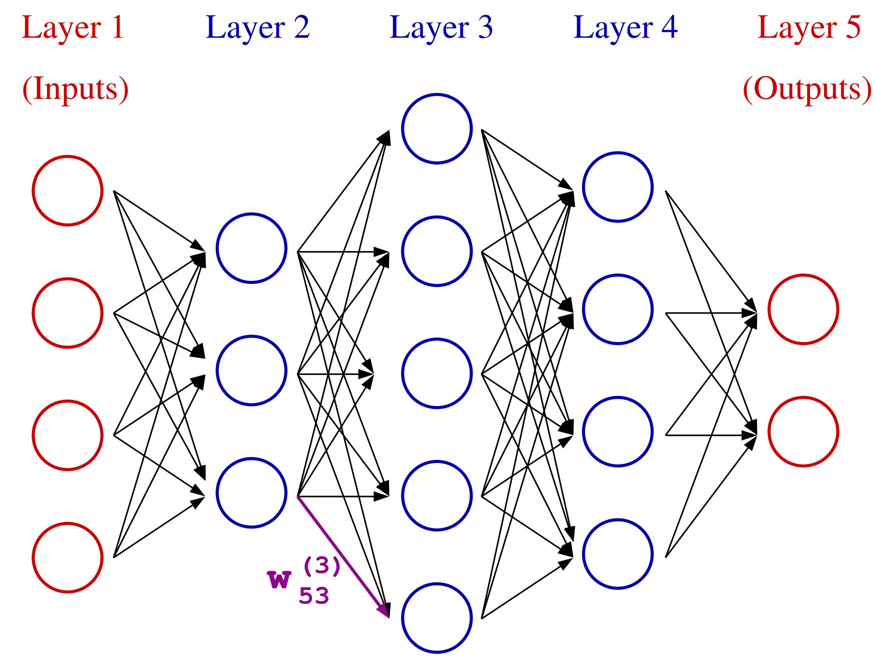
Training DNNs

Supervised training (GD)

Forward + Backward pass

Training 1 sample

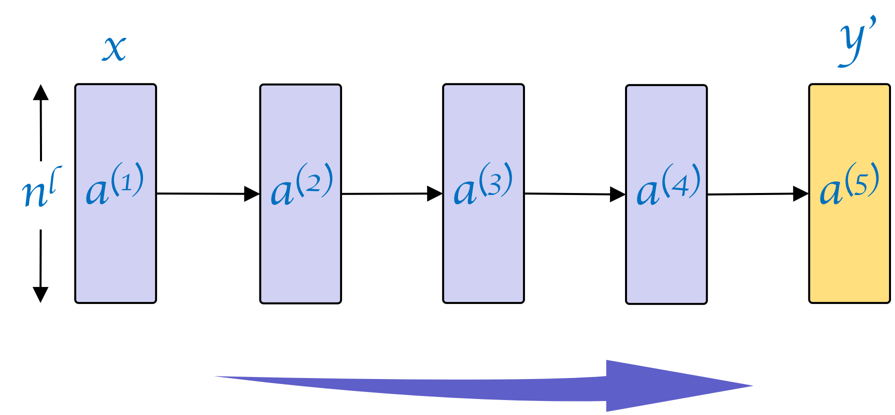
Assume given inputs x
and known outputs y



1. Forward pass (FP)

GEMV

$$\begin{aligned} a^{(1)} &= x, \\ a^{(l)} &= \sigma(z^{(l)}) = \sigma(W^{(l)}a^{(l-1)} + b^{(l)}) \in \mathbb{R}^{n_l}, \\ &\quad l = 2, 3, \ldots, L, \\ \tilde{y} &= a^{(L)}. \end{aligned}$$



Training DNNs

2. Backward pass (BP)

2.1. Gradient computation (GC)

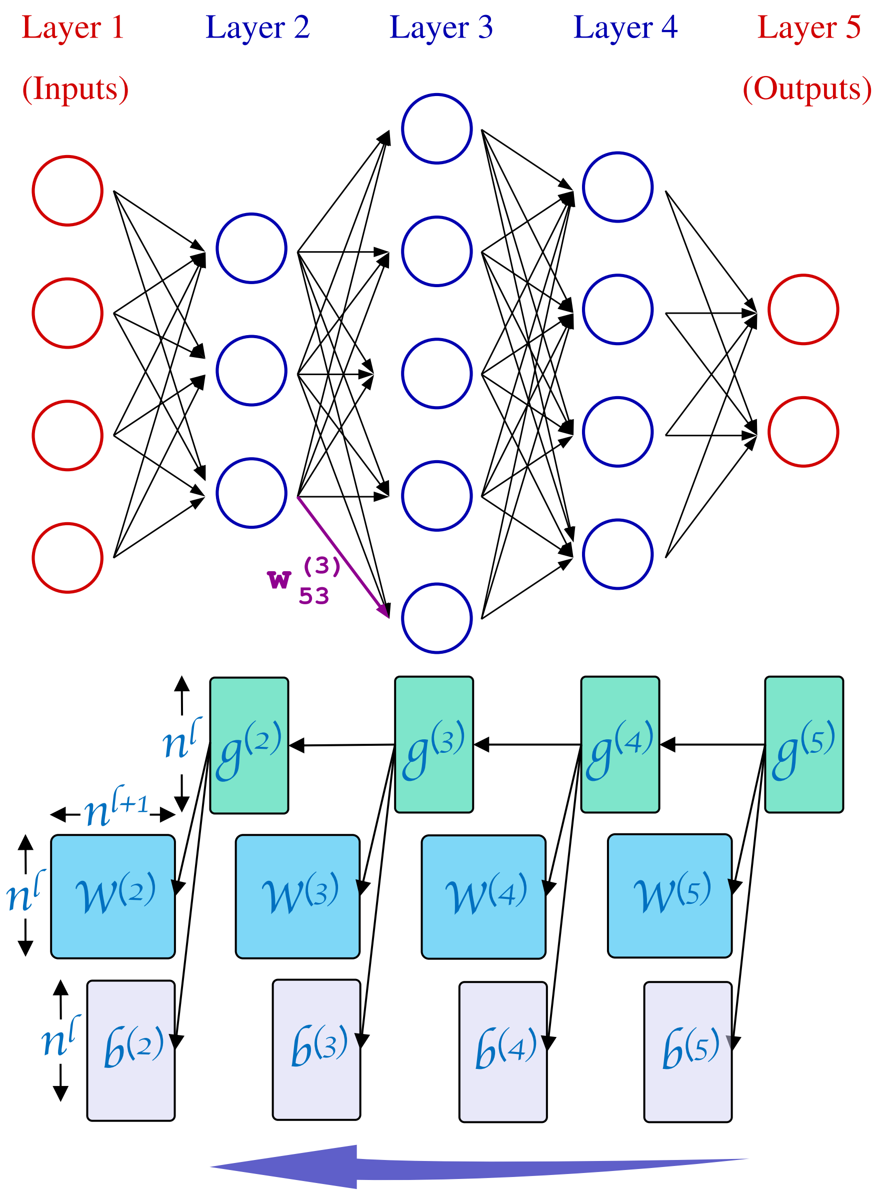
GEMV

$$
\begin{aligned}
g^{(L)} &= D^{(L)}(a^{(L)} - \tilde{y}), \\
g^{(l)} &= D^{(l)}(W^{(l+1)})^T g^{(l+1)}, \\
&\qquad l = L-1, L-2, \ldots, 2;
\end{aligned}
$$

2.2. Weight update (WU)

Outer-product

$$
\begin{aligned}
W^{(l)} &= W^{(l)} - \eta\, g^{(l)} (a^{(l-1)})^T, \\
b^{(l)} &= b^{(l)} - \eta\, g^{(l)}, \qquad l = L, L-1, \ldots, 2.
\end{aligned}
$$



High-performance Batched Training

From a single sample to a batch of b samples (SGD):

1. **Forward pass (FC)**

$$Z^{(l)} = W^{(l)} A^{(l-1)} + B^{(l)}$$ GEMM

2. **Backward pass**

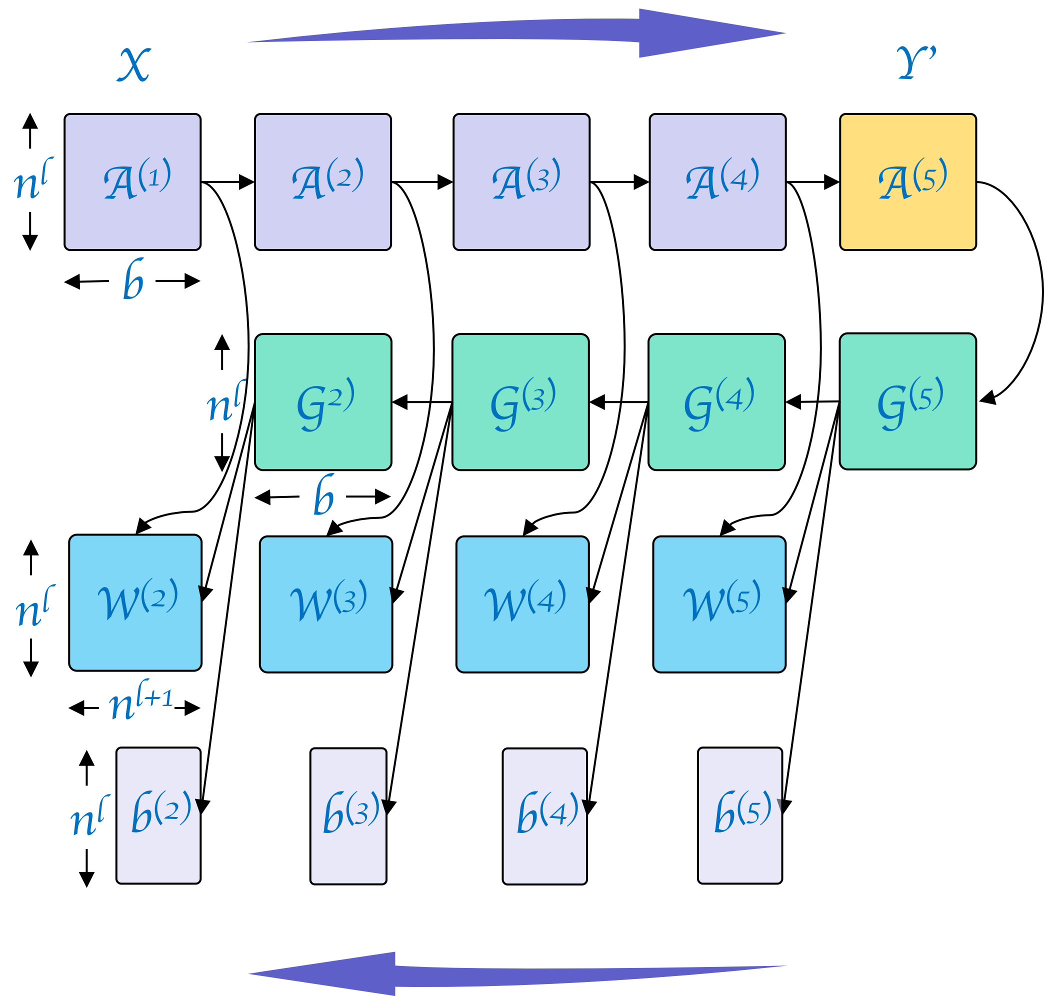
2.1. Gradient computation (GC)

$$G^{(l)} = (W^{(l+1)})^T G^{(l+1)}$$ GEMM

2.2. Weight update (WU)

GEMM

$$W^{(l)} = W^{(l)} - \eta \, G^{(l)} (A^{(l-1)})^T$$

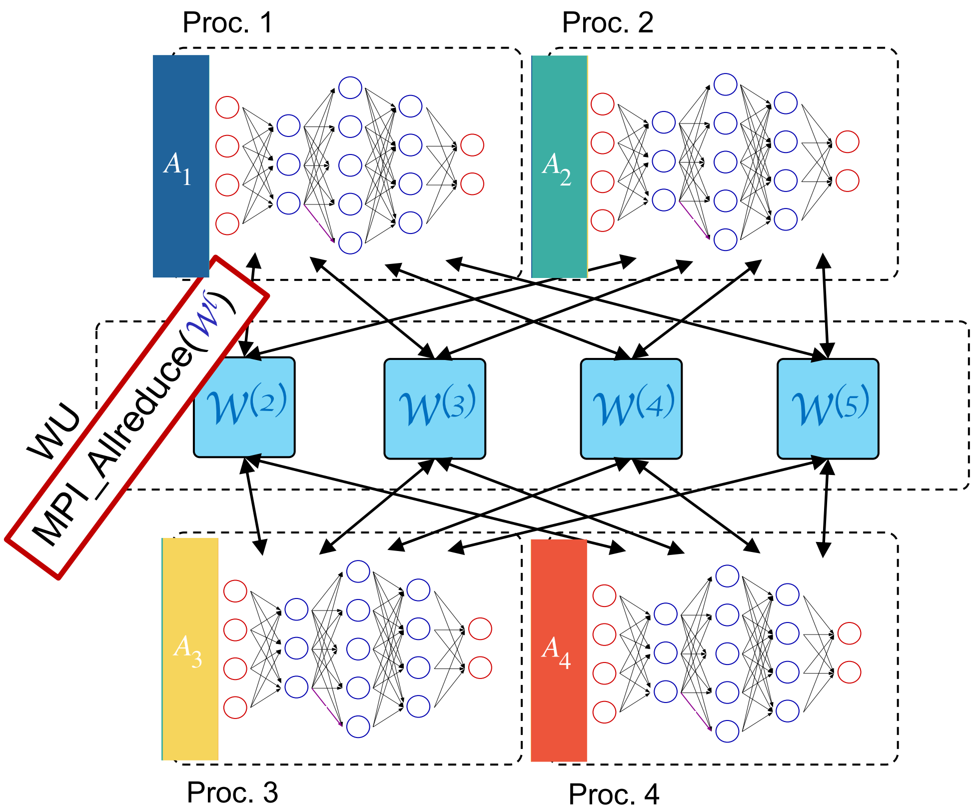


Index

1. Training DNNs
2. Parallel training on clusters
3. Performance model
4. Results
5. Conclusions

Parallel Training on Clusters

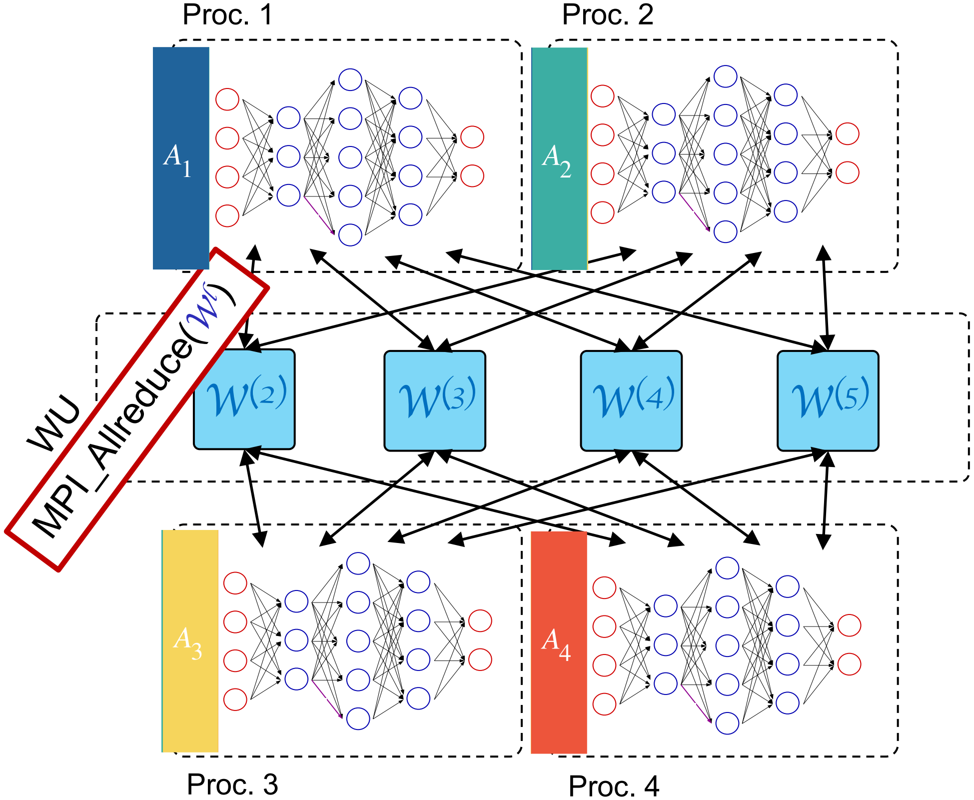
Data parallelism



- Batch size can be increased to feed all processes
- Scalability problems if model does not fit in memory

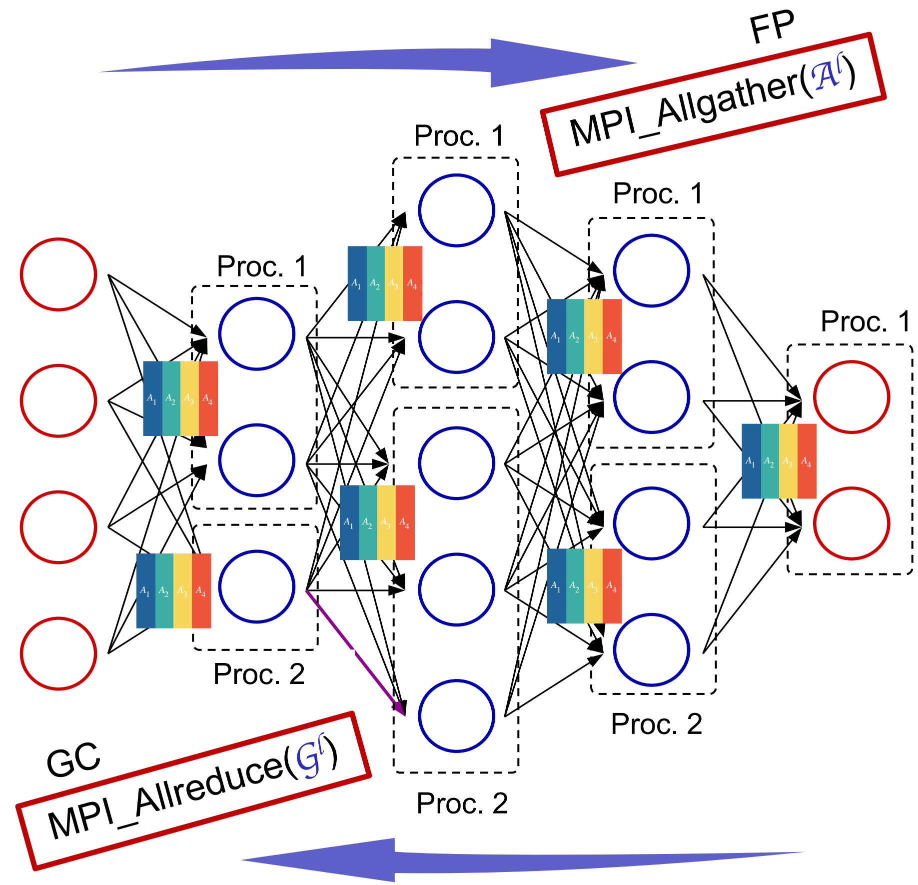
Parallel Training on Clusters

Data parallelism



- Batch size can be increased to feed all processes
- Scalability problems if model does not fit in memory

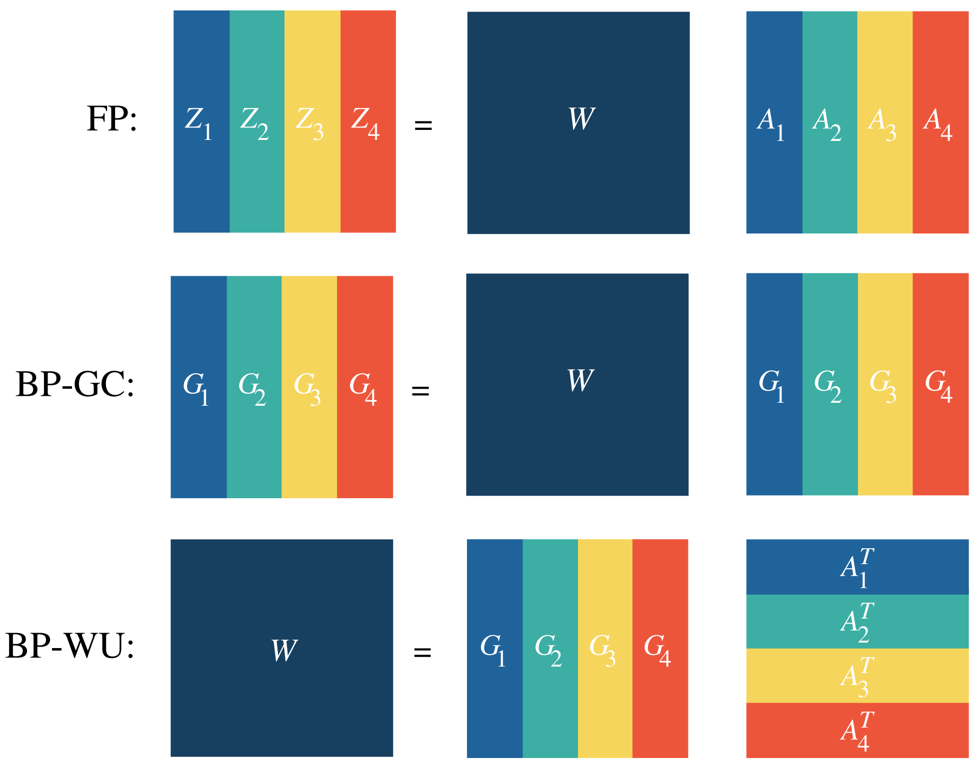
Model parallelism



- Suitable for large models
- More communications required (FP and GC)
- Increasing number of procs will not reduce runtime

Parallel Training on Clusters

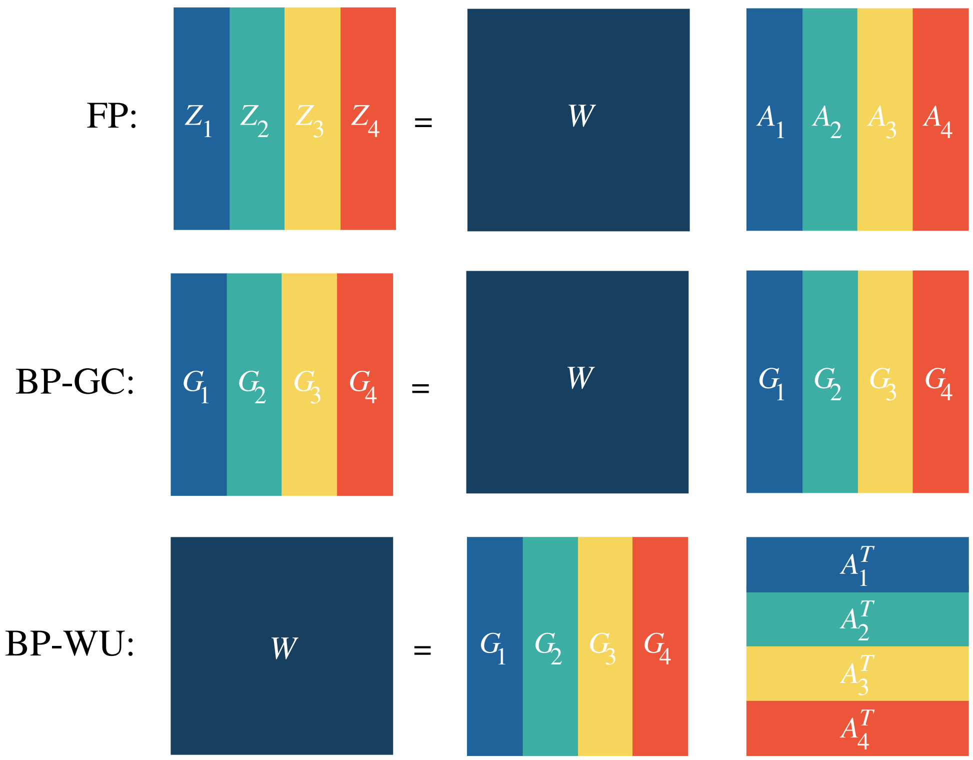
Data parallelism



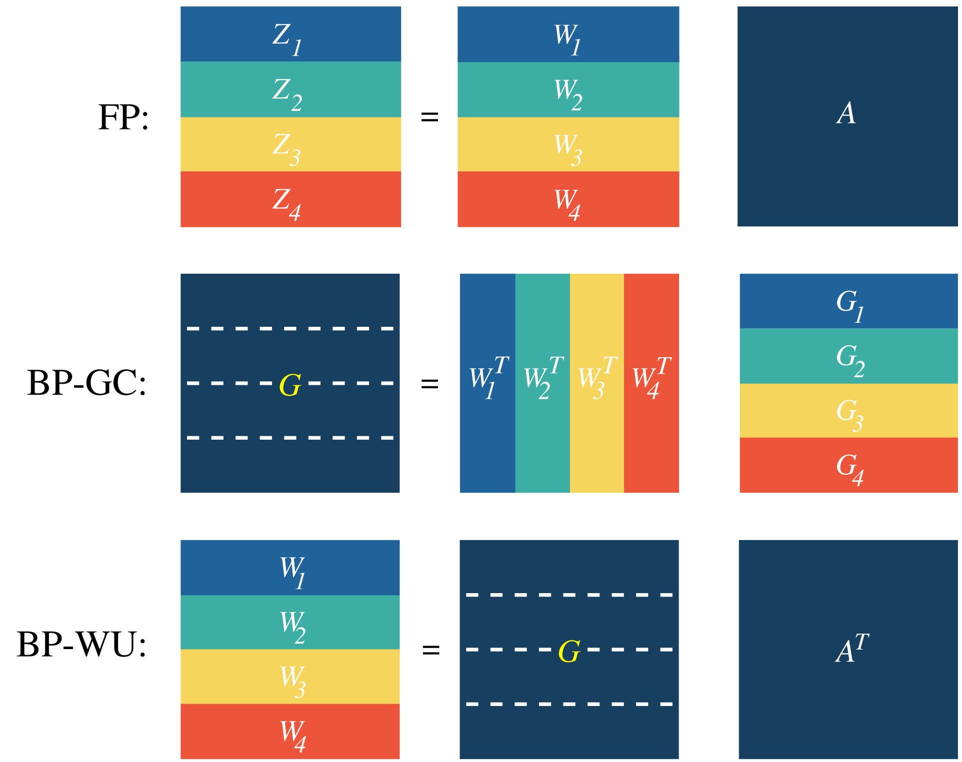
	C	A	B
FP	$n_l \times b$	$n_l \times n_{l-1}$	$n_{l-1} \times b$
BP-GC	$n_l \times b$	$n_l \times n_{l+1}$	$n_{l+1} \times b$
BP-WU	$n_l \times n_{l-1}$	$n_l \times b$	$b \times n_{l-1}$

Parallel Training on Clusters

Data parallelism



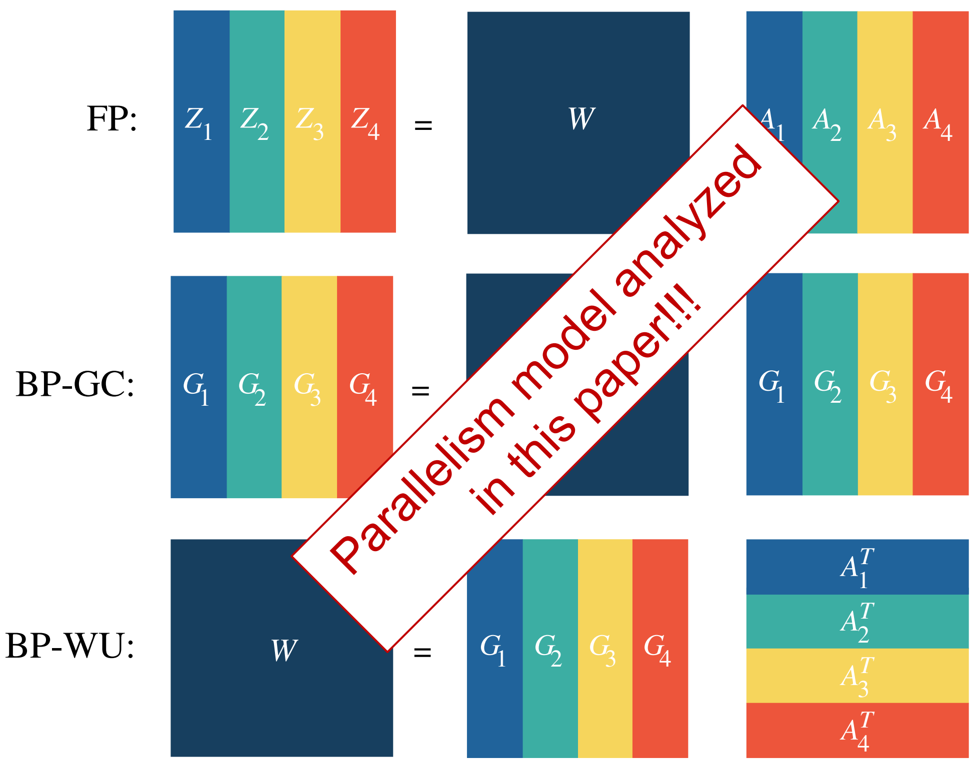
Model parallelism



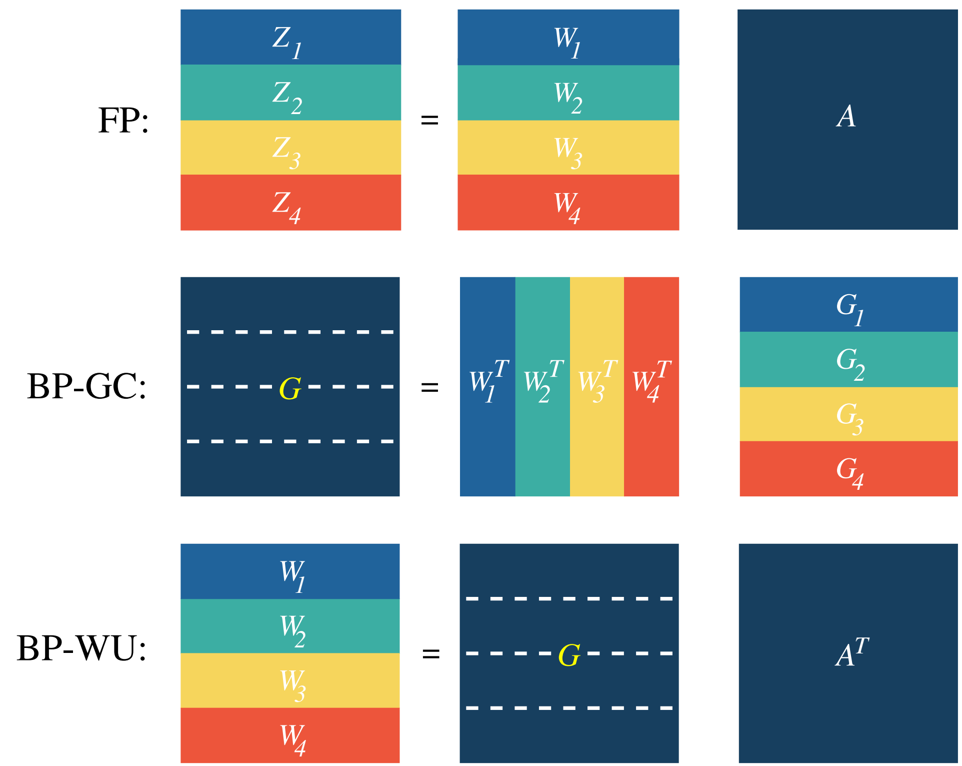
	C	A	B
FP	$n_l \times b$	$n_l \times n_{l-1}$	$n_{l-1} \times b$
BP-GC	$n_l \times b$	$n_l \times n_{l+1}$	$n_{l+1} \times b$
BP-WU	$n_l \times n_{l-1}$	$n_l \times b$	$b \times n_{l-1}$

Parallel Training on Clusters

Data parallelism



Model parallelism



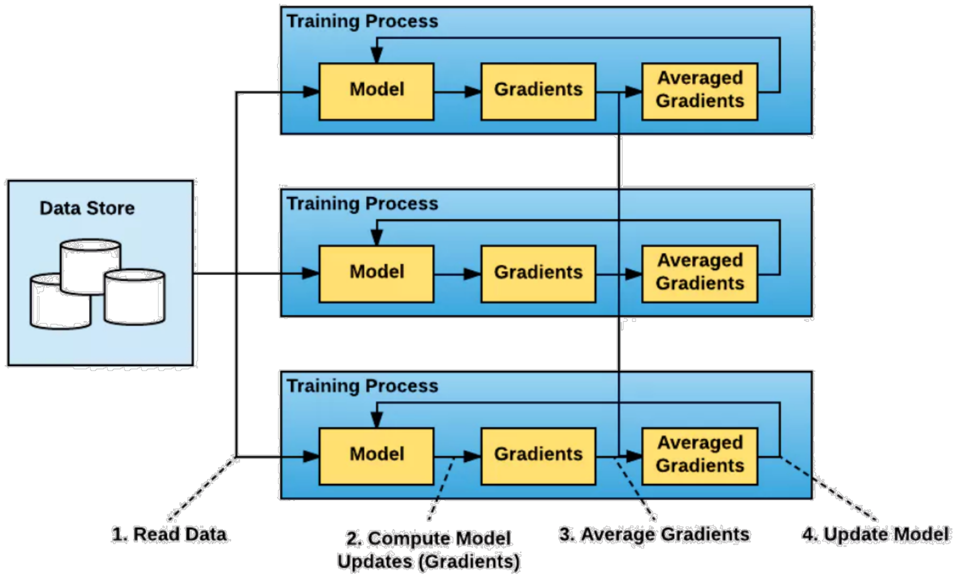
	C	A	B
FP	$n_l \times b$	$n_l \times n_{l-1}$	$n_{l-1} \times b$
BP-GC	$n_l \times b$	$n_l \times n_{l+1}$	$n_{l+1} \times b$
BP-WU	$n_l \times n_{l-1}$	$n_l \times b$	$b \times n_{l-1}$

Parallel Training in Clusters

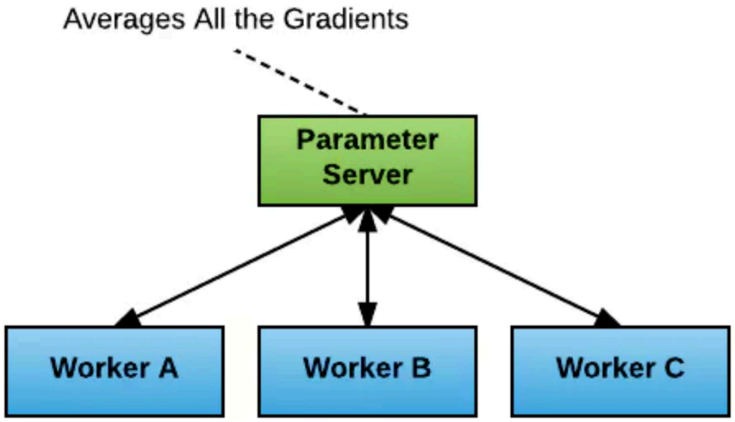
Data parallelism: Communication during BP

- For SGD, weight updates (gradients) must be communicated across all nodes

- Synchronous (e.g., Uber's Horovod)
  - Requires Allreduce operation



- Asynchronous (e.g., distributed TF)
  - Usually, via a parameter server which receives updates, aggregates them, and broadcasts them back



Index

1. Training DNNs
2. Parallel training on clusters
3. Performance model
4. Results
5. Conclusions

Performance Model

Model distributed data parallel training, considering:

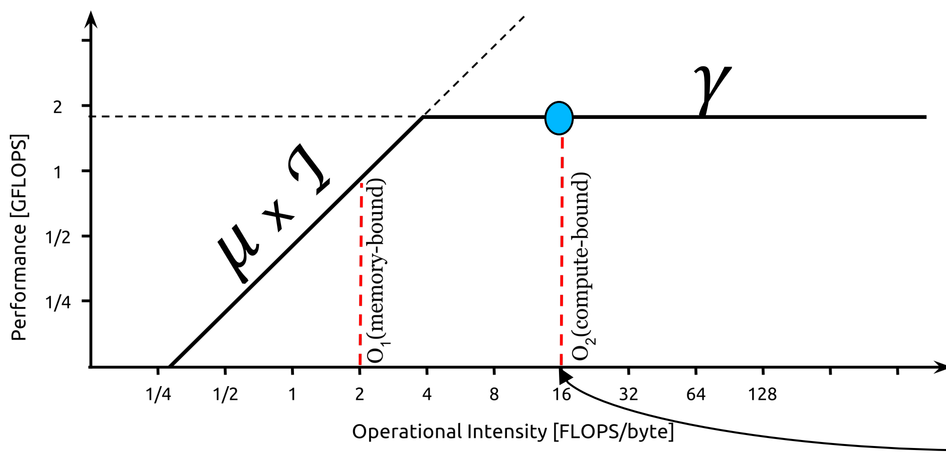
- Neural network model: CNNs of reference in our experiments
- Parallel cluster: $\mathcal{P}$ nodes offering γ FLOPS and μ bytes/s of mem. bandwidth
- Interconnect: Star (all-to-all) and 2D mesh cluster topologies with a latency of α and bandwidth of β bytes/s

Parameter	Meaning
L	Number of layers in the NN.
n_{l-1}, n_l	Number of inputs, outputs in layer l.
k_l^w, k_l^h	Kernel width, height of layer l (only CONV).
c_l	Number of kernels (channels) in layer l (only CONV).
b	Batch size.
δ	Bytes per floating-point number.
P	Number of nodes in the parallel cluster.
γ	Theoretical peak performance (in FLOPS).
μ	Memory bandwidth (in bytes/s).
α	Link latency (in s).
β	Link bandwidth (in bytes/s).

Performance Model

Node performance:

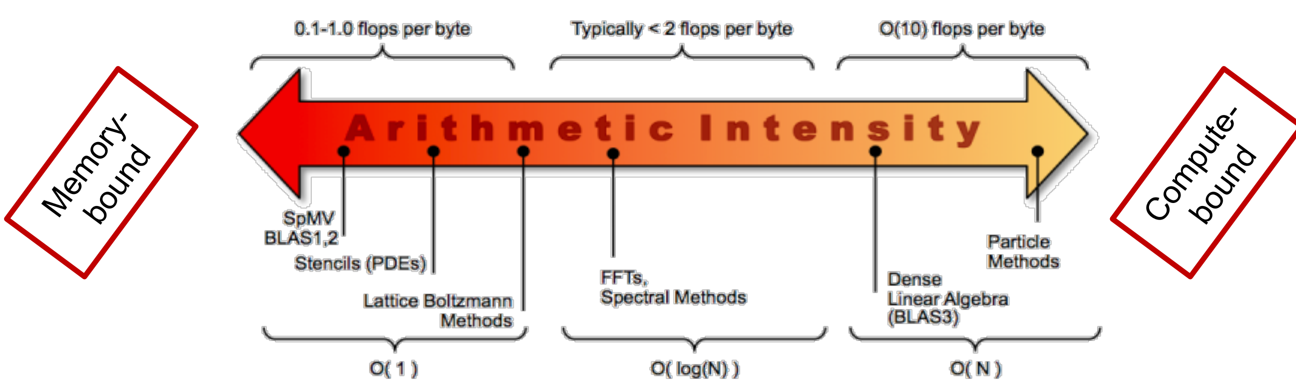
- Roofline model: calculate performance by using the arithmetic intensity



Example: GEMM

$$I_{\text{FP/FC}} = \frac{2 n_l n_{l-1} b}{(n_l b + n_{l-1} b + n_l n_{l-1})\delta} \text{ flops/byte}$$

Estimated execution time = flops / min(μ x $\mathcal{I}$, GFLOPS)

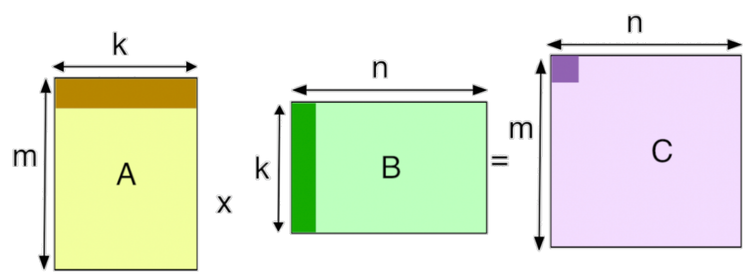


Performance Model

Node performance:

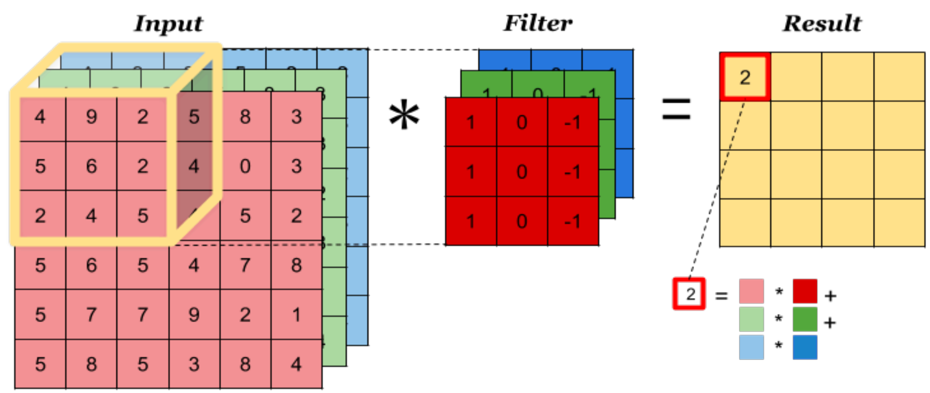
- <u>Roofline model</u>: flops ($2mnk$) / memops ($2mn+mk+nk$) for FC and CONV

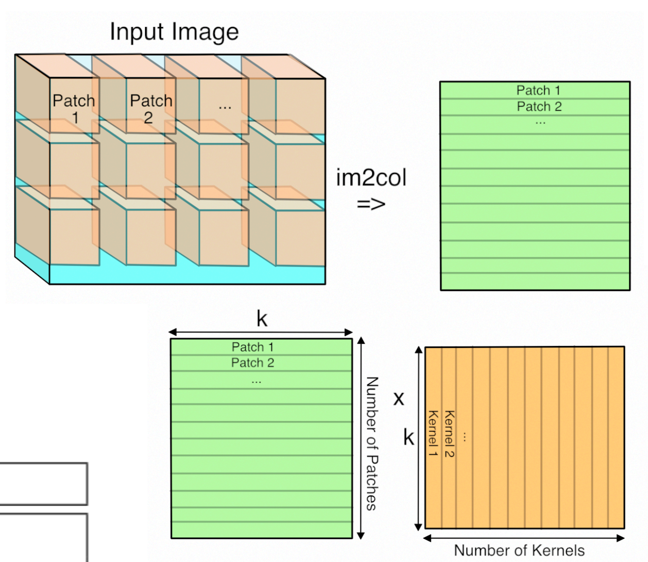
FC



	C	A	B
FP	$n_l \times b$	$n_l \times n_{l-1}$	$n_{l-1} \times b$
BP-GC	$n_l \times b$	$n_l \times n_{l+1}$	$n_{l+1} \times b$
BP-WU	$n_l \times n_{l-1}$	$n_l \times b$	$b \times n_{l-1}$

CONV





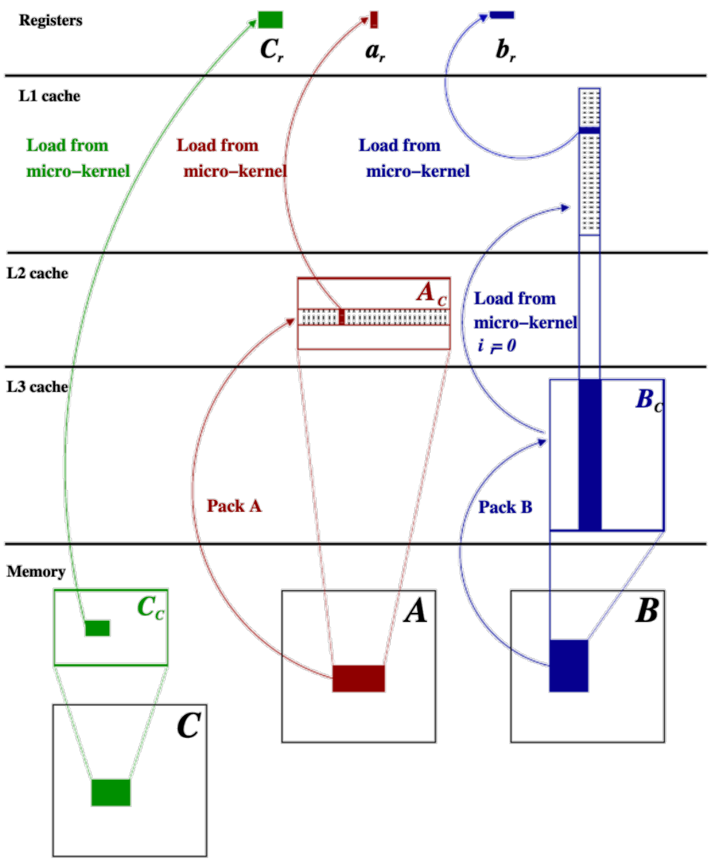
	C	A	B
FP	$b \times c_l \times h_l \times w_l$	$b \times c_{l-1} \times h_{l-1} \times w_{l-1}$	$c_l \times c_{l-1} \times h'_l \times w'_l$
BP-GC	$b \times c_l \times h_l \times w_l$	$b \times c_{l+1} \times h_{l+1} \times w_{l+1}$	$c_l \times c_{l+1} \times h'_{l+1} \times w'_{l+1}$
BP-WU	$c_l \times c_{l-1} \times h'_l \times w'_l$	$b \times c_{l-1} \times h_{l-1} \times w_{l-1}$	$b \times c_l \times h_l \times w_l$

Performance Model

Node performance:

- We consider a blocked implementation of GEMM, as it is done in GotoBLAS2, OpenBLAS, BLIS, or Intel MKL
- Partitioning according to a specific level of cache hierarchy (OpenBLAS and BLIS)
  - m, n, k are reduced to m_c, n_c, k_c so that B_c fits in L3, A_c fits in L2, and C to RAM

for $j_c = 0, 1, \ldots, n-1$ **in steps of** n_c
  for $p_c = 0, 1, \ldots, k-1$ **in steps of** k_c
    $B(p_c : p_c + k_c - 1, j_c : j_c + n_c - 1) \rightarrow B_c$
    for $i_c = 0, 1, \ldots, m-1$ **in steps of** m_c
      $A(i_c : i_c + m_c - 1, p_c : p_c + k_c - 1) \rightarrow A_c$
      // Macro-kernel
      $C_c(i_c : i_c + m_c - 1, j_c : j_c + n_c - 1)$
       $+= A_c(i_c : i_c + m_c - 1, p_c : p_c + k_c - 1)$
       $\cdot\ B_c(p_c : p_c + k_c - 1, j_c : j_c + n_c - 1)$
    endfor
  endfor
endfor



- Intensity is then computed as:

$$I_{\mathrm{FP/FC}} = \frac{2 m_c n_c k_c}{(m_c n_c + m_c k_c + k_c n_c)\delta}$$

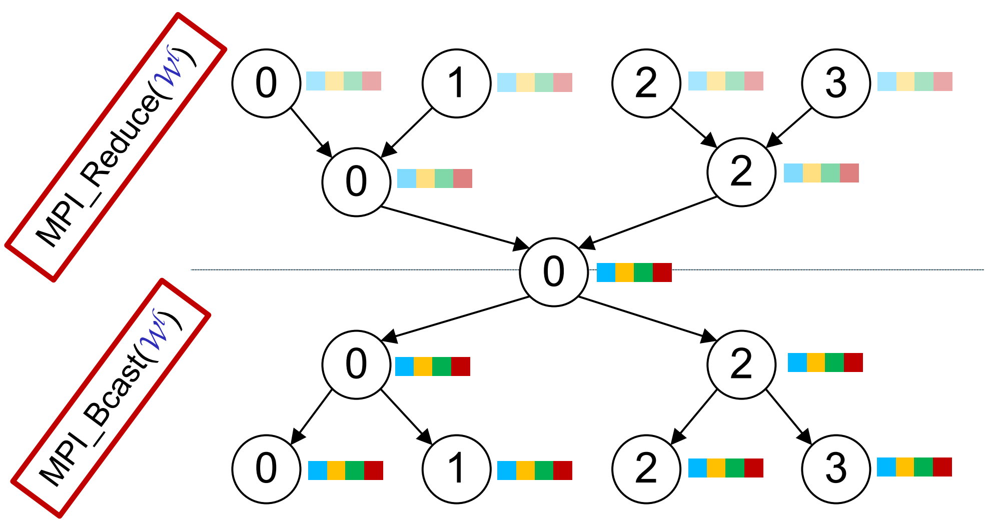
Performance Model

Network transmission performance (MPI_Allreduce):

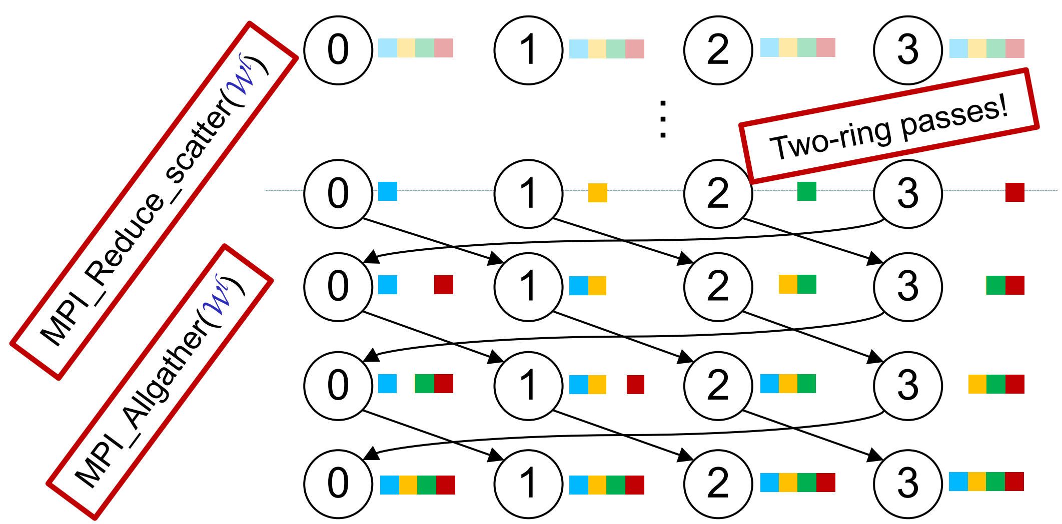
- <u>Allreduce</u>: Minimum spanning tree (MST) and Bucket (BKT)

Topology	Star		Mesh	
Algorithm	Latency	Communication	Latency	Communication
MST	$2\lceil \log_2 P \rceil (\alpha_1 + \alpha_3)$	$2\lceil \log_2 P \rceil \frac{s\delta}{\beta}$	$2\sum_{k=0}^{d-1} \lceil \log_2 d_k \rceil (\alpha_1 + \alpha_3)$	$2\sum_{k=0}^{d-1} \lceil \log_2 d_k \rceil \frac{s\delta}{\beta}$
BKT	$2P\alpha_1$	$2\frac{(P-1)}{P}\frac{s\delta}{\beta}$	$2\sum_{k=0}^{d-1} d_k \alpha_1$	$2\frac{(P-1)}{P}\frac{s\delta}{\beta}$

<u>MST</u> (Reduce + Broadcast)



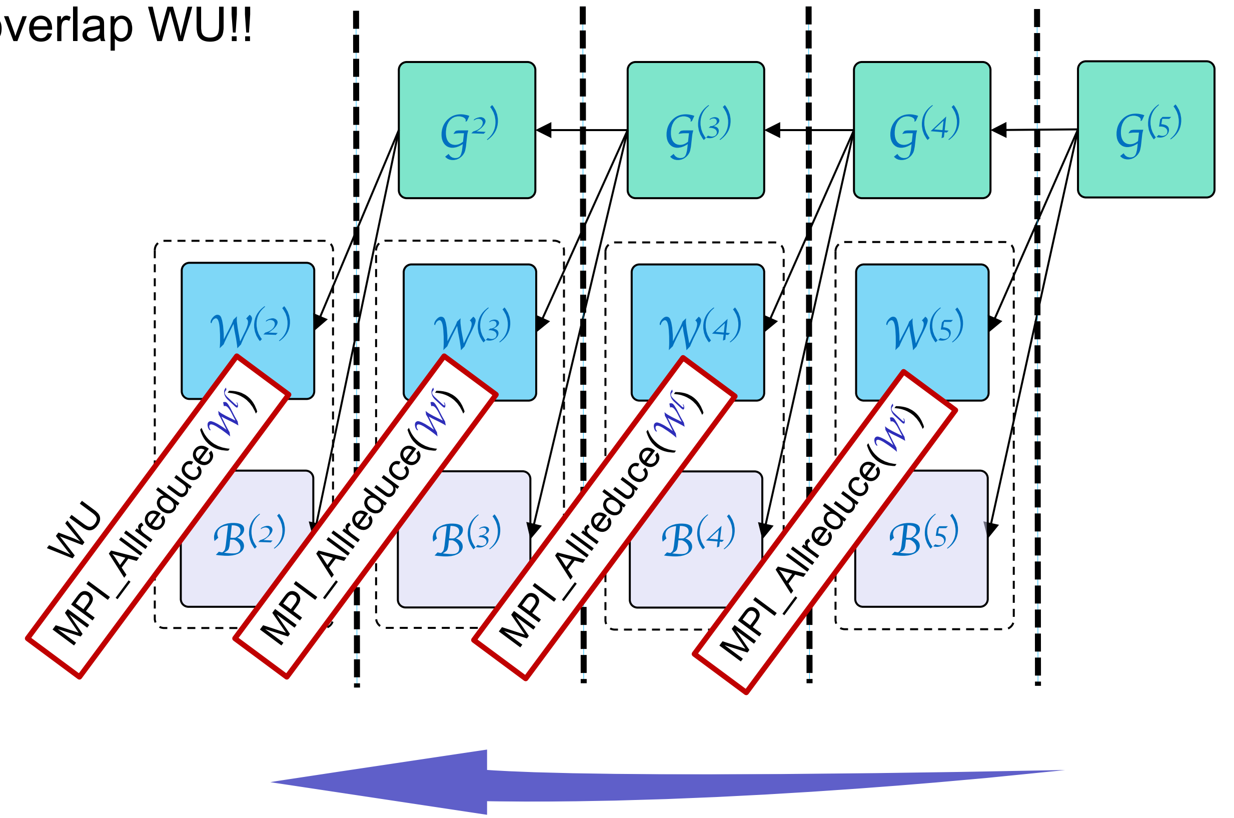
<u>BKT</u> (Reduce-Scatter + Allgather)



Performance Model

Overlapping communication with computation in BP:

- Strict data dependencies: FP: $\mathcal{A}^{(2)} \rightarrow \mathcal{A}^{(2)} \rightarrow \ldots\ \mathcal{A}^{(\mathcal{L})}$; GC: $\mathcal{G}^{(\mathcal{L})} \rightarrow \mathcal{G}^{(\mathcal{L}\text{-}1)} \rightarrow \ldots\ \mathcal{G}^{(2)}$
- But MPI_Allreduce($\mathcal{W}^{(l)}$) can be performed in parallel with $\mathcal{G}^{(\mathcal{L}\text{-}1)}, \mathcal{G}^{(\mathcal{L}\text{-}2)}, \ldots, \mathcal{G}^{(2)}$
  - GC can overlap WU!!



Index

1. Training DNNs
2. Parallel training on clusters
3. Performance model
4. Results
5. Conclusions

Results

CNN models:

Model	FC	CONV	POOL	Total
AlexNet	3	5	3	11
Inception v3	1	94	14	109
ResNet-50 v2	1	53	1	55
VGG16	3	13	5	21

SKYLAKE cluster parameters:

Parameters	SKYLAKE cluster
# of Nodes	1,000
Interconnect	Dual-rail Mellanox EDR Infiniband
Link bandwidth (Gbps)	200
Max. link latency (μs)	0.5

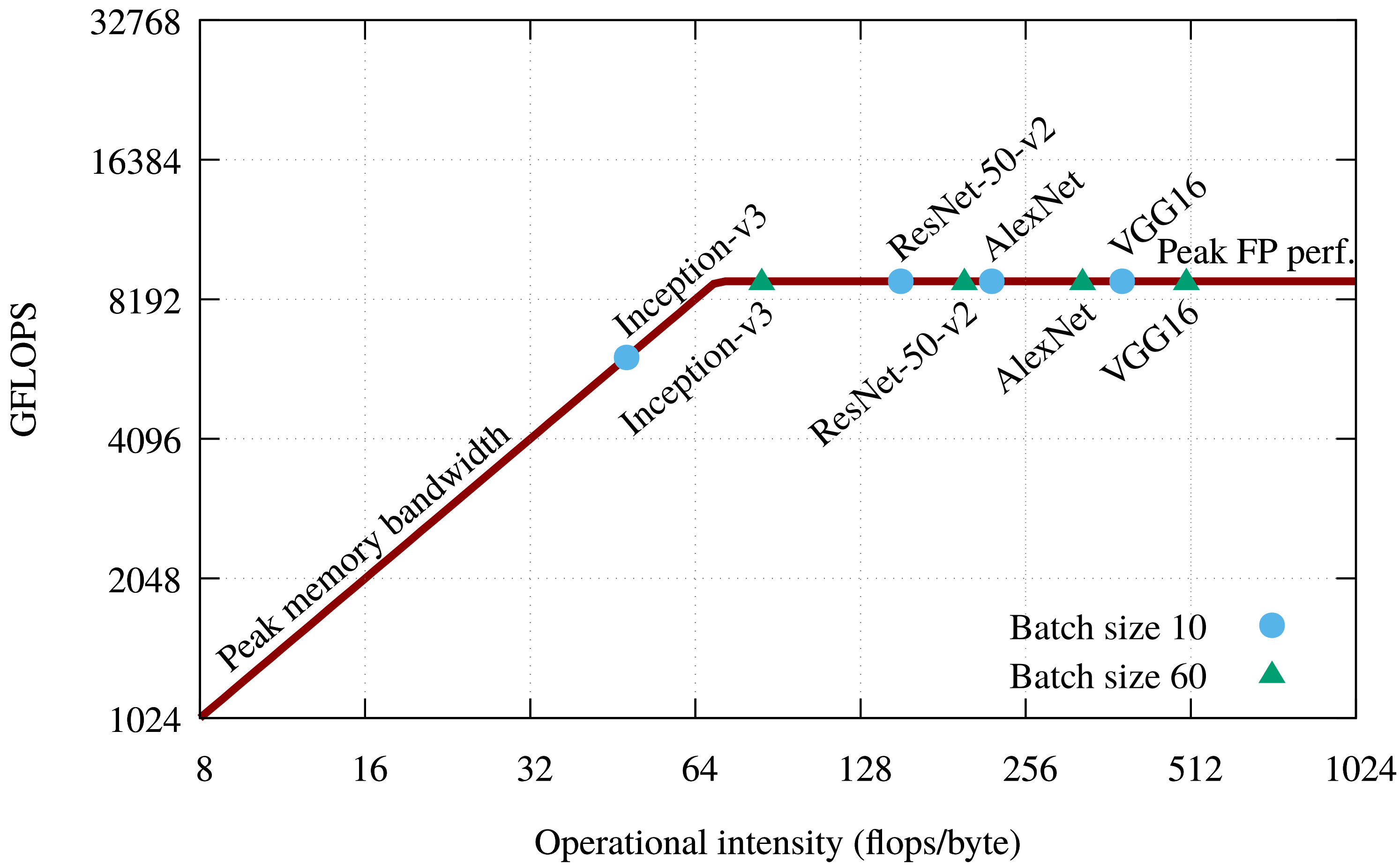
SKYLAKE node parameters:

Parameters	SKYLAKE node
Processor model	Intel Xeon Platinum 8180M
Max. FP32 throughput (flops/cycle)	64
Frequency (GHz)	2.5
# of Cores	28 (56 2-way SMT)
Peak FP32 performance (GFLOPS)	8,960
Mem. bus width (Bytes)	8
Mem. clock rate (GHz)	2.666
Mem. channels	6
Peak mem. bandwidth (GBytes/s)	128
DDR4 RAM memory (GBytes)	256

BLIS GEMM on a SKYLAKE Intel processor: m_c^o = 480, n_c^o = 3072, and k_c^o = 384

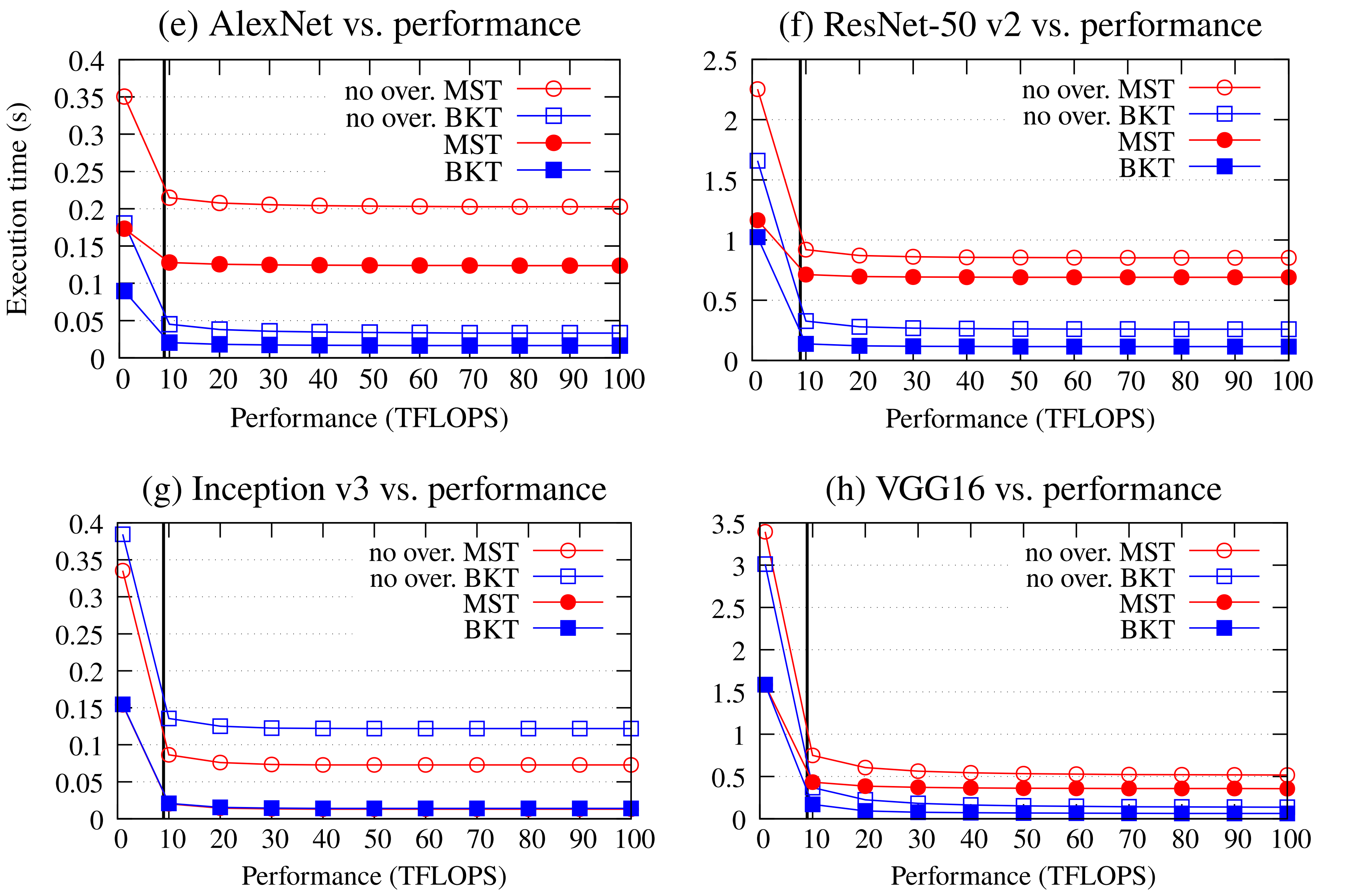
Arithmetic intensity of CNNs

- Batch size 10 and 60 samples per process, i.e., 10k and 60k samples in total

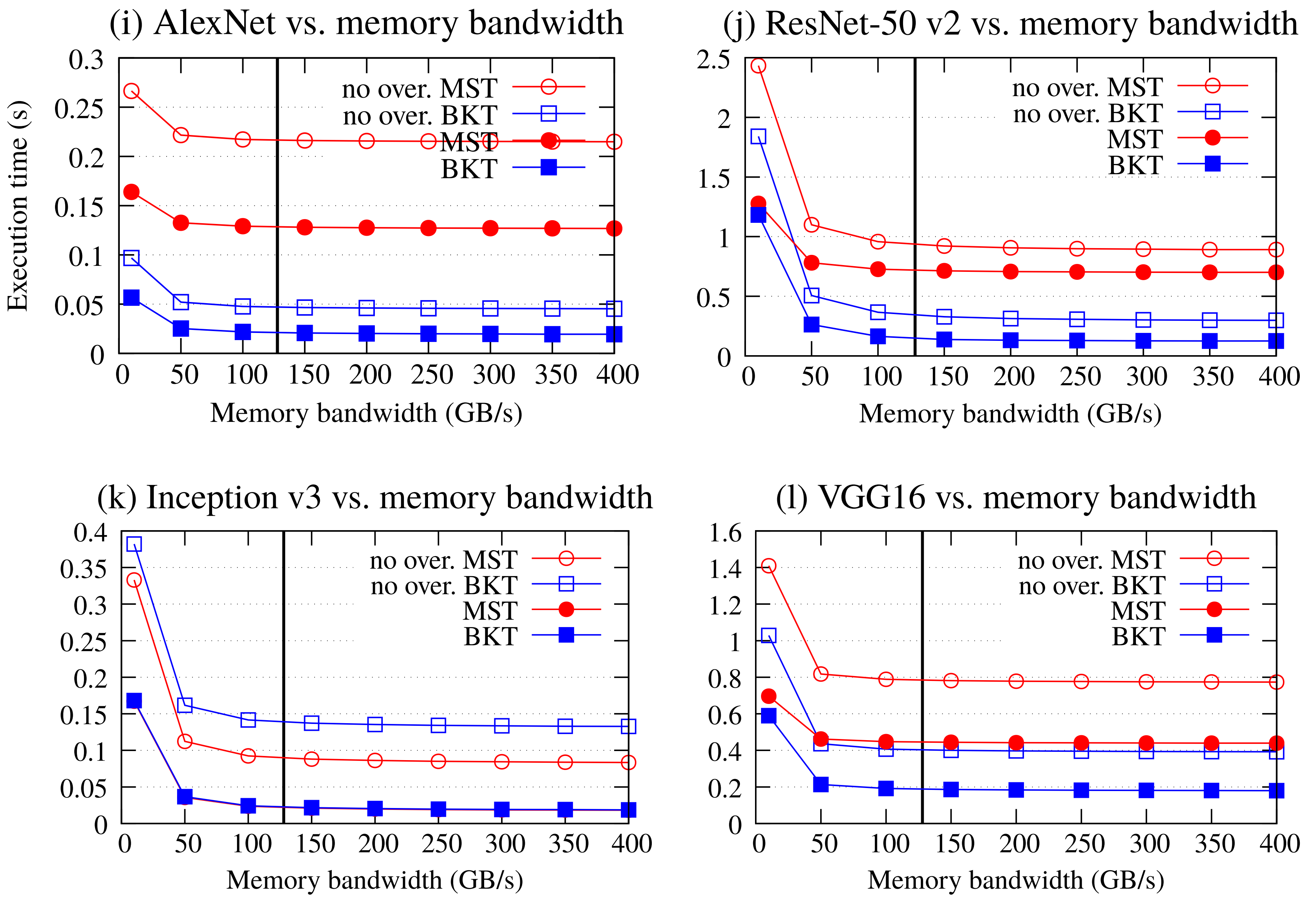


Most CNNs are compute bound, except for Inception-v3 with batch size 10k

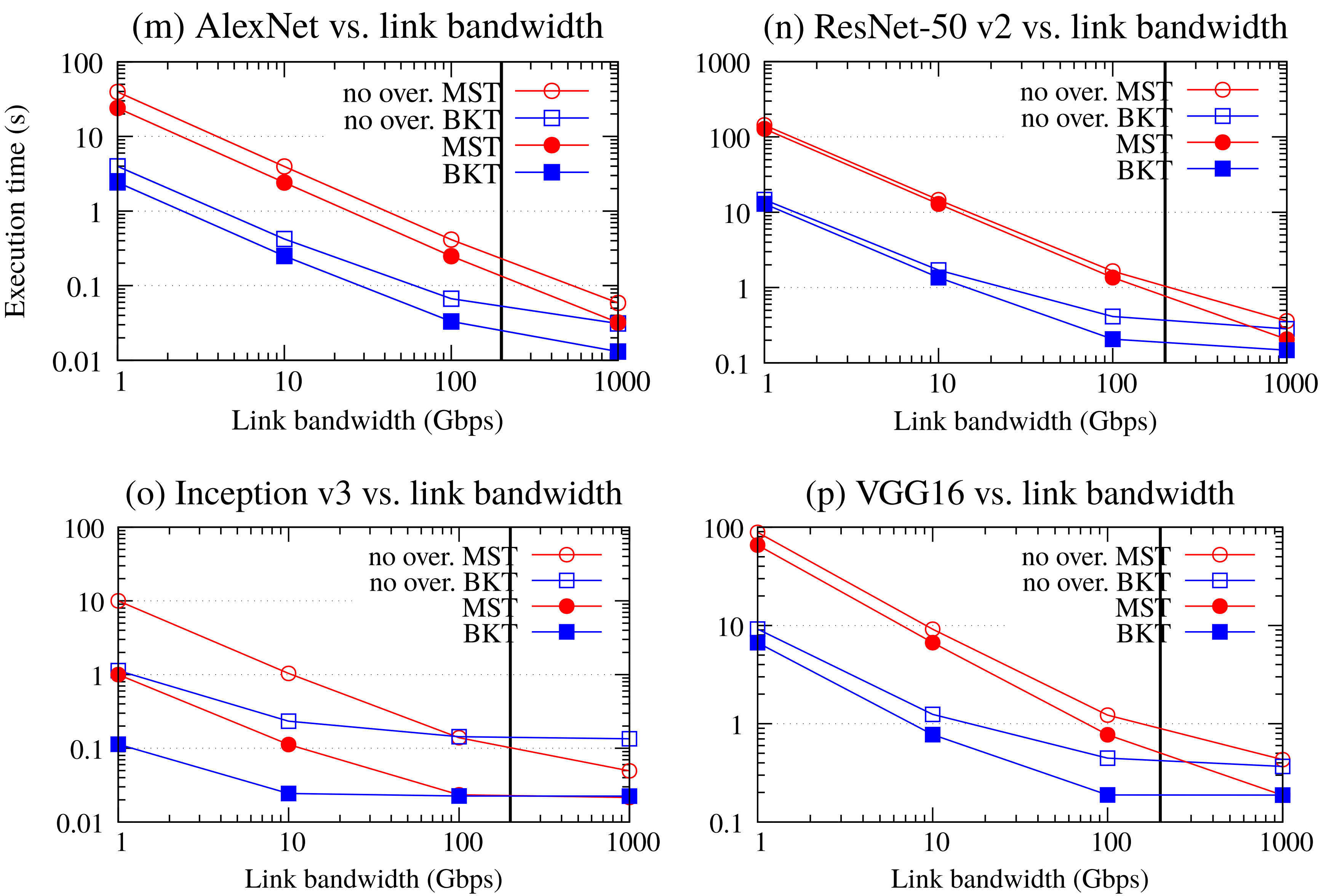
Performance of distributed training



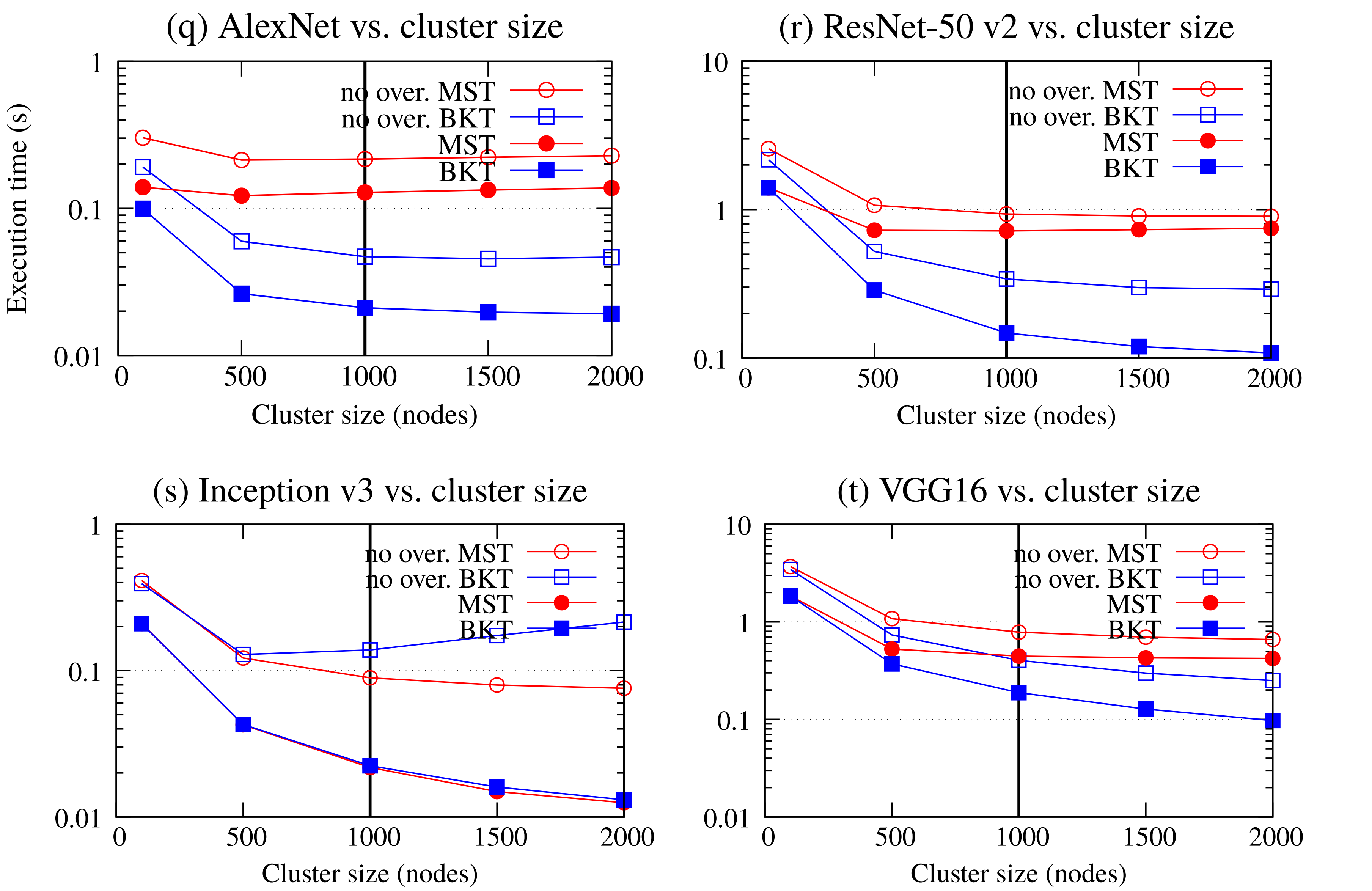
Performance of distributed training



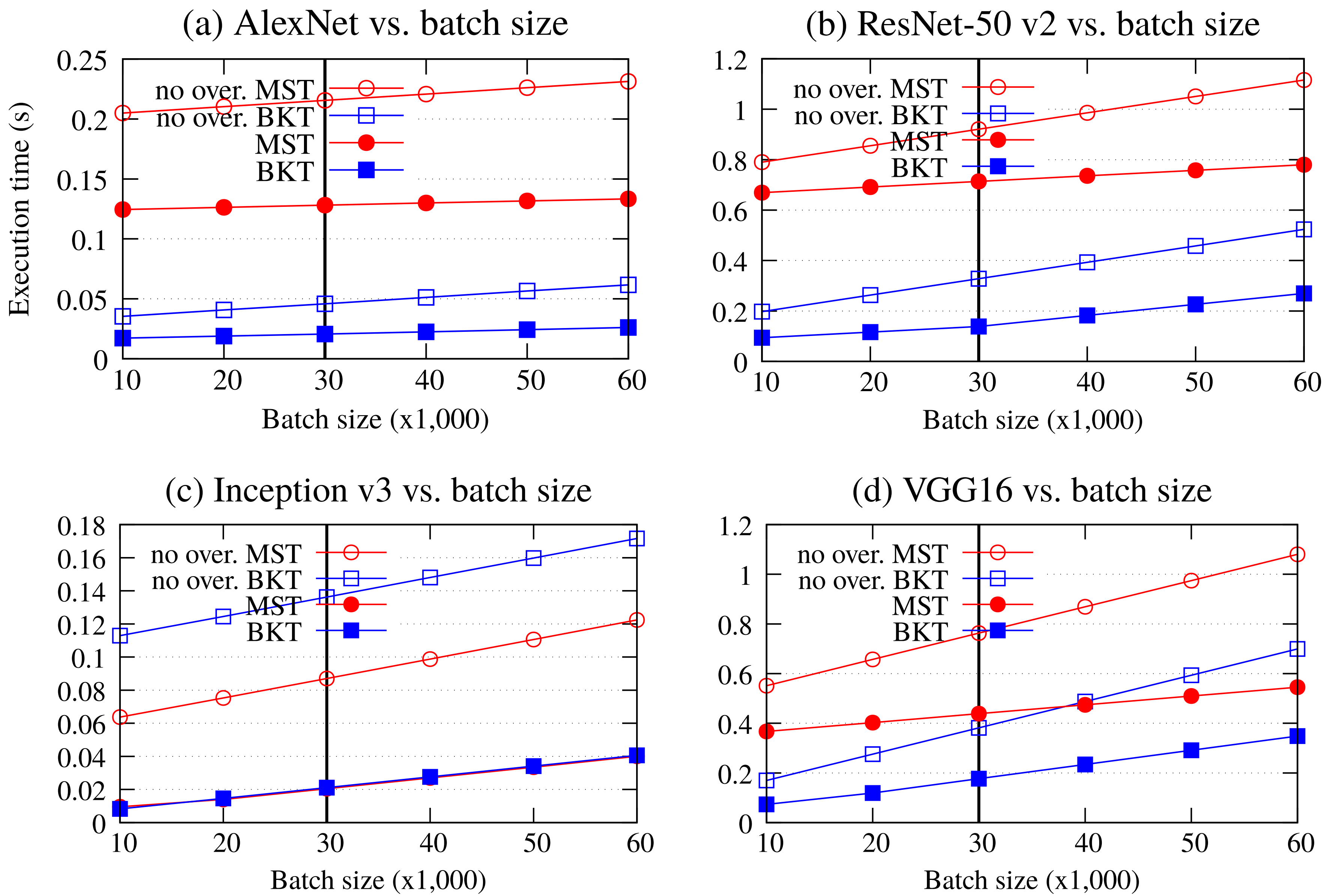
Performance of distributed training



Performance of distributed training



Performance of distributed training



Index

1. Training DNNs
2. Parallel training on clusters
3. Performance model
4. Results
5. Conclusions

Conclusions

- Current CNN models are mainly compute-bound:
  - Improving the processor capabilities does not bring noticeable improvements
- Overlapping communication with computation increases the performance!
- Network is important: increasing link bandwidth may reduce the execution time of collective operations
- Larger mini-batches provide a more accurate estimate of the gradients and accelerate the optimization process, however each step requires longer computations
  - However, increasing mini-batch size can hurt test accuracy!

Thank you for your attention!

Questions?