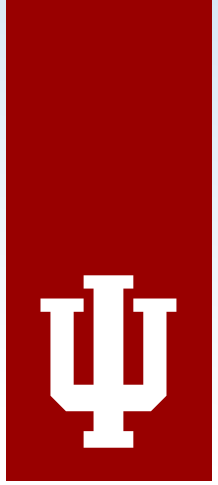


Flexible Communication for Optimal Distributed Learning over Unpredictable Networks

Sahil Tyagi and Martin Swany

Introduction

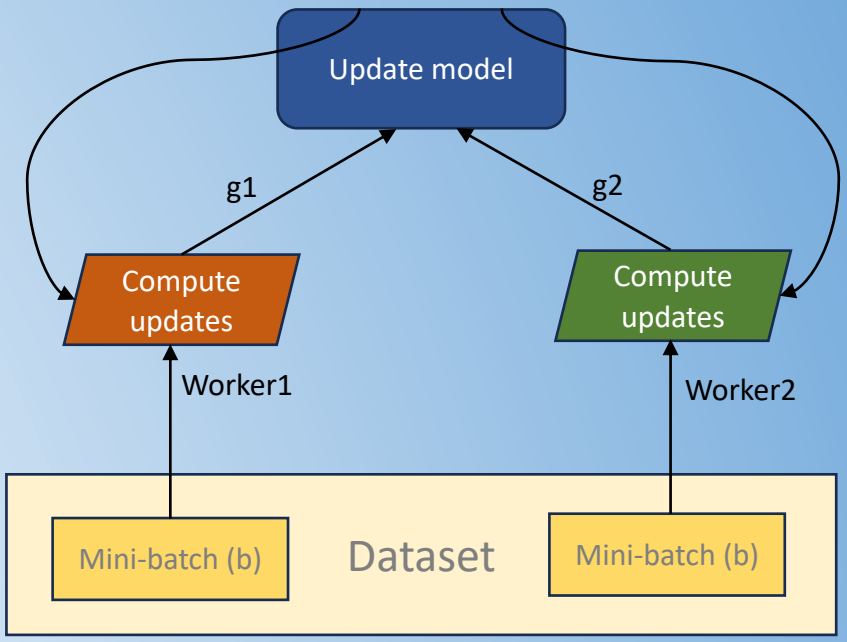
Network Fluctuations and Variability

- Unpredictable network performance prevalent across all edge, cloud and HPC
- Cloud VMs avail different tiers of networking bandwidth based on pricing
- In federated learning, edge devices have slower networks than fog nodes
- Network heterogeneity in data-centers arises from device placement, topology, link b/w
- Network bandwidth may fluctuate over time because of:

Network Congestion | QoS Prioritization | Resource Contention | Network Scheduling

Distributed Data-Parallel Training

- Computational requirements for SOTA DL training doubles every **3.4 months**
- Computation overhead is proportional to model size
- Parallel performance improved by performing ***more work per-iteration***



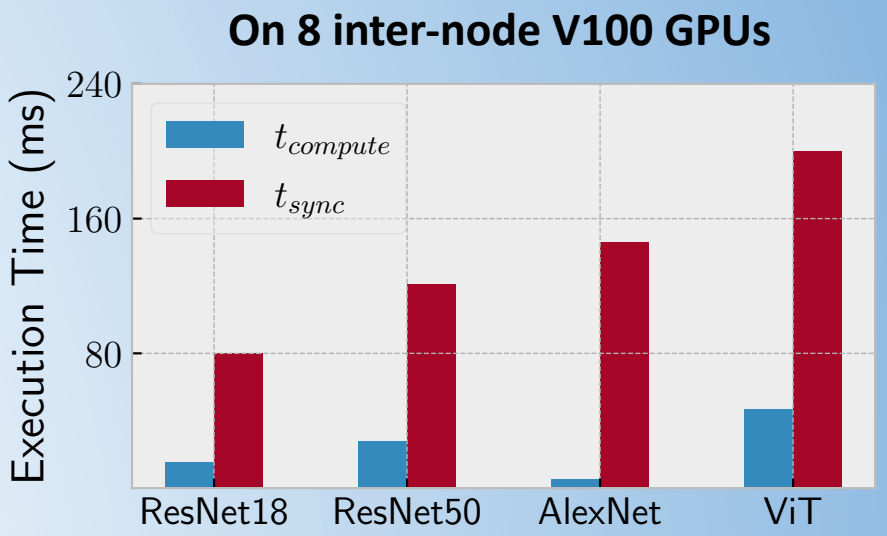
$$w_{i+1} = w_i - \eta \frac{1}{N} \sum_{n=1}^{n=N} \frac{\partial}{\partial w_i} \left(\frac{1}{|b|} \sum_{d_{(i,n)} \in \mathcal{D}_n} \mathcal{L}(x_{(i,n)}, w_i) \right)$$

Parallel Scaling in Distributed Learning

- DNN training is an iterative-convergent process; each training step is *computationally identical*

$$t_{step} = t_{compute} + t_{sync} + t_{IO}$$

- Aggregating updates among workers is ***communication-heavy***, limiting linear scaling of distributed training jobs



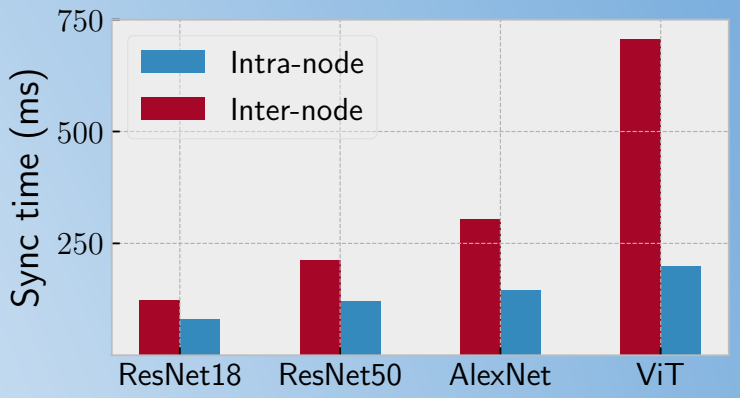
Parallel Scaling in Distributed Learning cont'd

- Communication cost is influenced by worker placement in the cluster
- Synchronous training aggregates updates either in a *centralized* or *decentralized* topology
- Based on the alpha-beta communication model, collectives have varying overheads

$$\alpha \rightarrow \text{Latency} \qquad \frac{1}{\beta} \rightarrow \text{Bandwidth}$$

$$M \rightarrow \text{Message-size} \qquad N \rightarrow \#\ \text{workers}$$

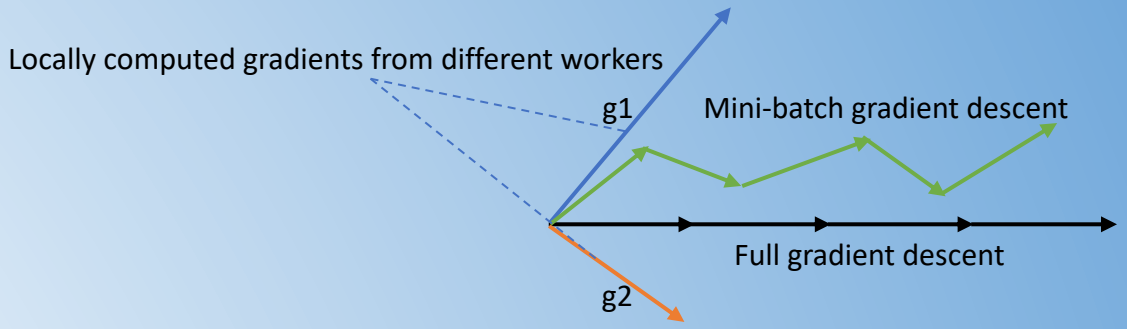
8 GPUs over PCIe vs. 10Gbps network



Operation	Communication cost
PS (Star)	$2\alpha + 2(N-1)M\beta$
Ring-AllReduce	$2(N-1)\alpha + 2\frac{(N-1)}{N}M\beta$
Tree-AllReduce	$2\alpha\log(N) + 2\log(N)M\beta$

Statistical Aspect of Data-Parallel Scaling

- Although DL training is iterative-convergent, features learned over the epochs are not *equally* attributed to each iteration



- This is due to the stochastic nature of gradient descent; ***iterations don't contribute equally towards overall model learning***

- Gradients computed from a randomly sampled mini-batch are ***noisier*** compared to full-batch gradient descent

- Aggregated gradients are a better approximation of the true gradients computed over entire training data

Background and Related Work

Communication scheduling

- Synchronization cost hidden via computation-communication overlap; gradients of out layer are transmitted while inner layers are still processing
- Can be inefficient on models with numerous layers of non-uniform size
- *Tensor-fusion* and *gradient bucketing* combines tensors to perform efficient reduction
- However, the scope of interleaving communication with compute diminishes on newer hardware

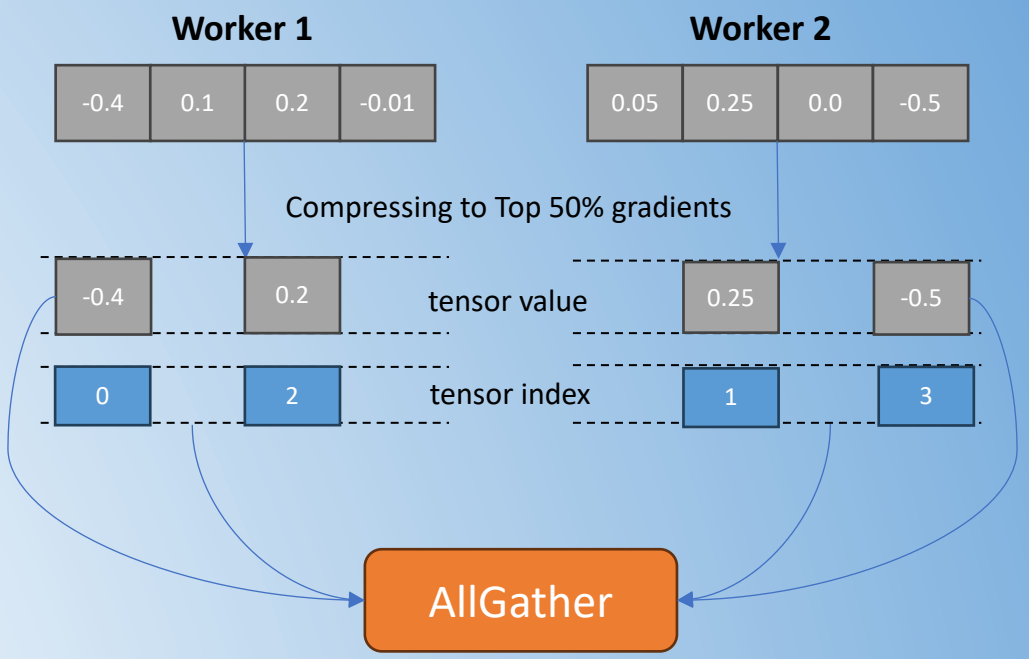
Gradient Compression

- Sparse methods like Top-*k* reduce communication volume by choosing largest *k*% gradients

- Methods like LWTop-*k* and MSTop-*k* apply different threshold estimation mechanisms and communicate top k% gradient values + indices

- Per-iteration cost in gradient compression is comprised of:

$$t_{step}^{(c)} = t_{compute} + t_{sync}^{(c)} + t_{IO} + t_{comp-decomp}^{(c)}$$



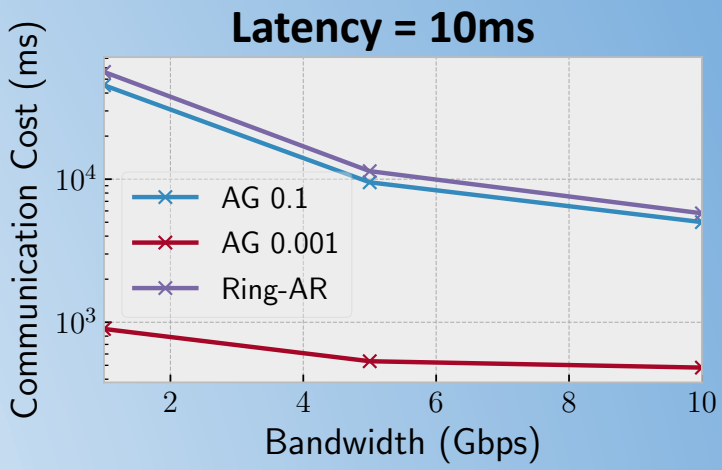
For compression to be viable over DenseSGD

$$t_{sync}^{(c)} + t_{comp-decomp}^{(c)} < t_{sync}$$

Gradient Compression vs. DenseSGD

- DenseSGD can perform reduction via send/recv or AllReduce

Operation	Communication cost
PS (Star)	$2\alpha + 2(N-1)M\beta$
Ring-AllReduce	$2\alpha\log(N) + 2\log(N)M\beta$
Tree-AllReduce	$2(N-1)\alpha + 2\frac{(N-1)}{N}M\beta$
AllGather	$\alpha\log(N) + (N-1)M\beta$
Broadcast	$\alpha\log(N) + \log(N)M\beta$



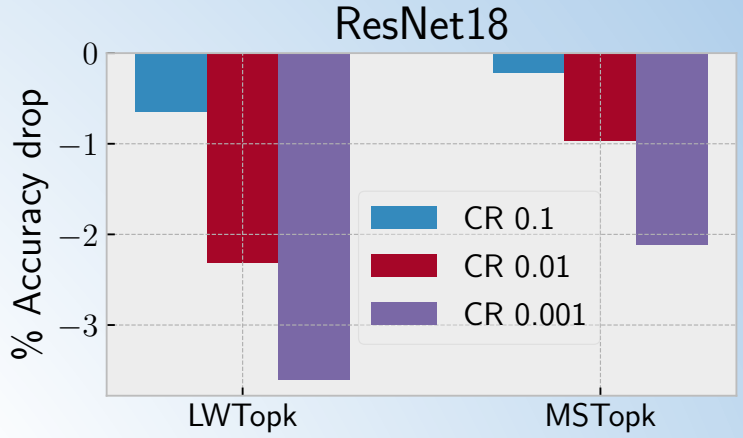
- We vary latency-bandwidth via traffic control ('*tc*') over 1B tensor and compare communication + compression overhead at various CRs

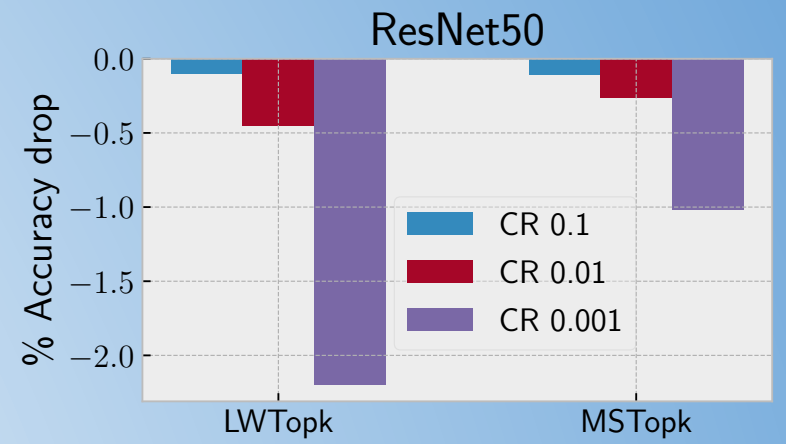
Statistical Efficiency of Gradient Compression

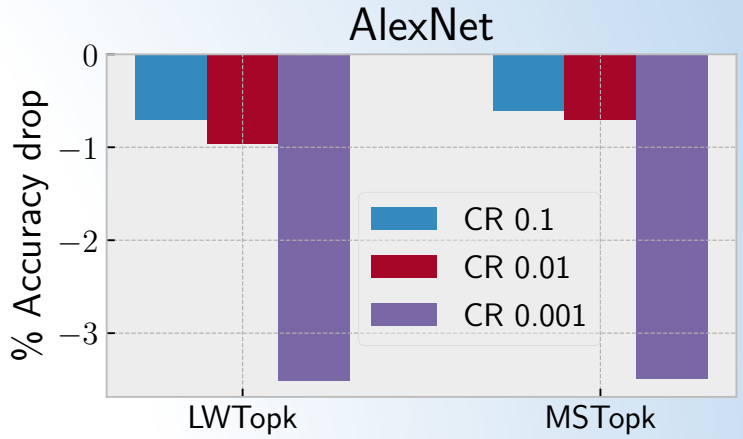
High CRs reduce communication cost, lower iteration time and improve speedup

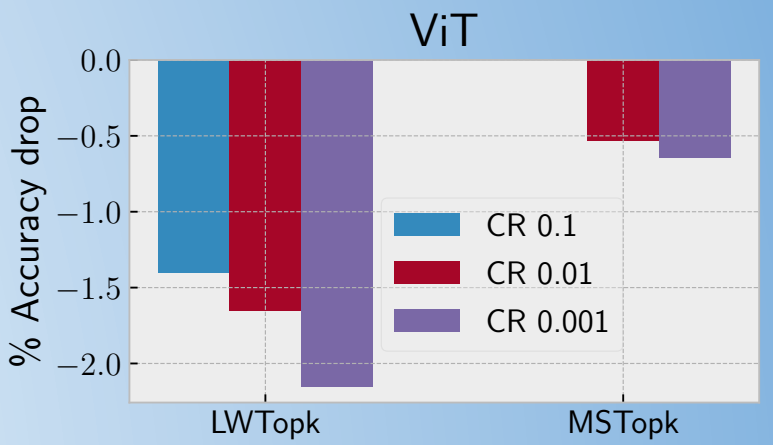
But higher CRs tend to degrade final model quality as well!

We compare final model accuracy of LWTop*k* and MSTop*k* compression at various CRs









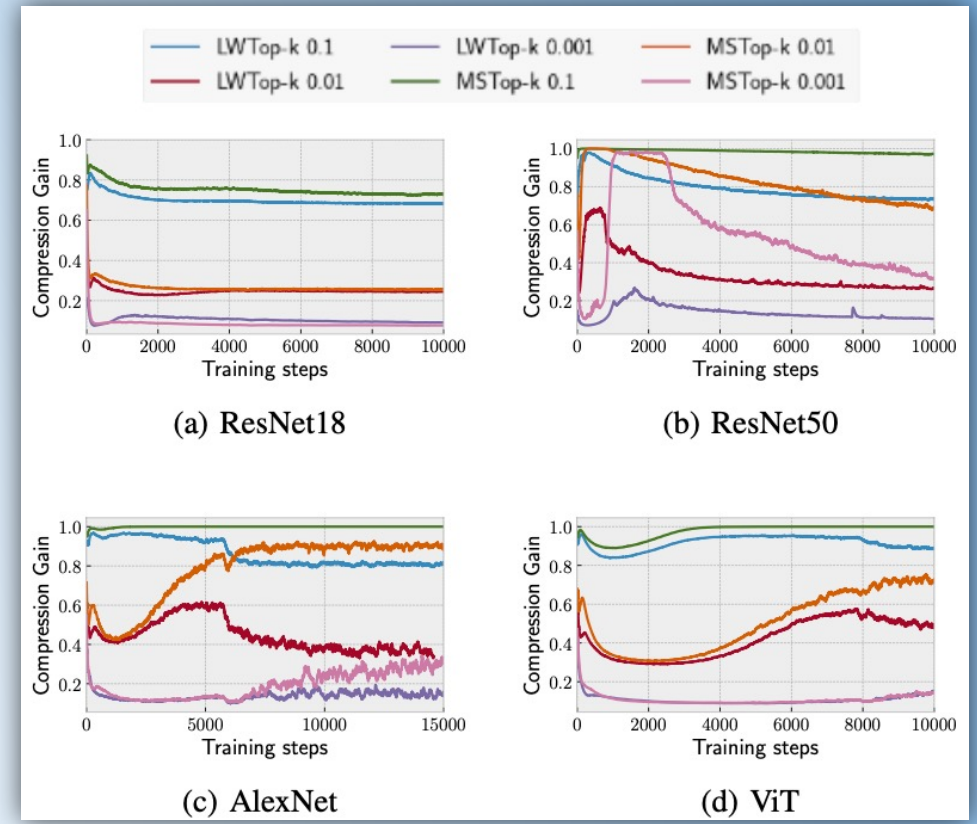
Statistical Efficiency of Gradient Compression cont'd

- With high CRs, compression loses significant gradient information that ***degrades accuracy***
- Mitigated via *error-feedback* mechanism

$$g_e^{(i)} = g_o^{(i)} + \texttt{residual}^{(i-1)}$$
$$g_c^{(i)} = \mathcal{C}(g_e^{(i)}) \quad \text{and} \quad \texttt{residual}^{(i)} = g_e^{(i)} - g_c^{(i)}$$

- Statistical efficiency in compression measured by comparing variance in prior and post compression gradients

$$\text{Compression Gain} = \frac{\mathrm{E}[||g_c^{(i)}||^2]}{\mathrm{E}[||g_e^{(i)}||^2]}$$



AR-Top*k* Compression

Compression with the Optimal Collective

- Optimal communication collective between AllReduce variants (Ring, Tree, etc.) or AllGather depends on model-size, cluster-size, target CR (if any) and network configuration (e.g., topology, latency, bandwidth)

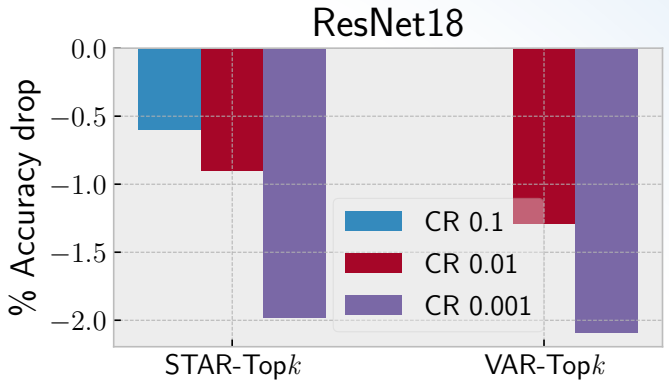
- For cases where AllReduce has lower communication overhead, we propose AllReduce friendly Top-*k* compression-communication mechanism called **AR-Top*k***

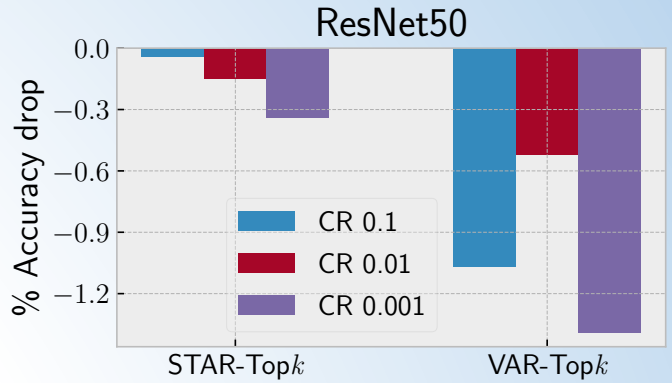
- In AR-Top*k*, one worker *broadcasts* its indices across the cluster; Gradients at broadcasted indices are then aggregated across via *AllReduce*

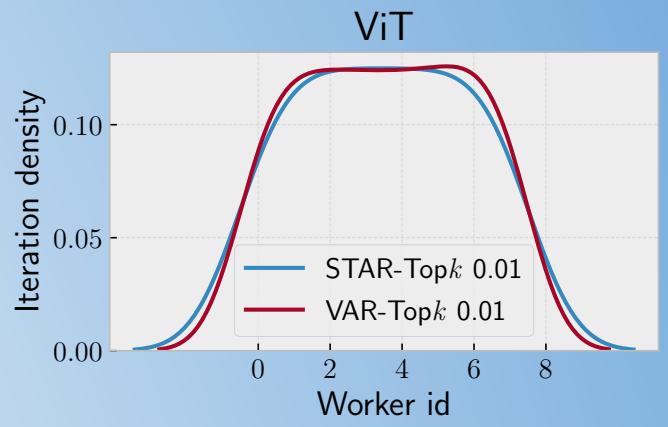
How to choose which worker broadcasts its indices to all other workers in the cluster to aggregate its top-*k* gradients?

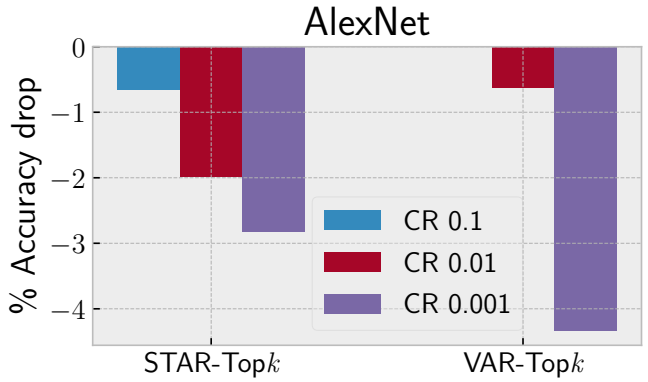
Worker selection in AR-Top*k* Compression

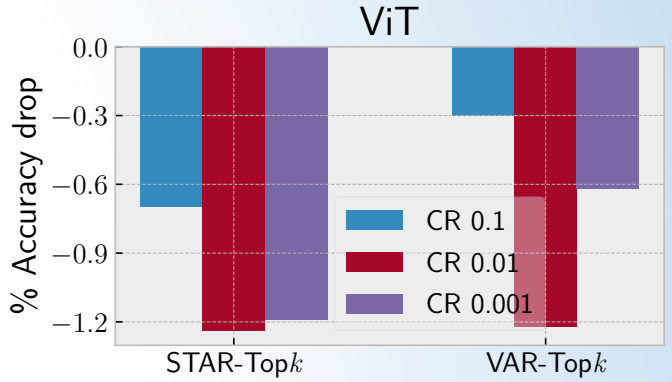
- Worker can be selected in a round-robin manner (**STAR-Top*k***) or based on the worker with most volatile updates (**VAR-Top*k***)

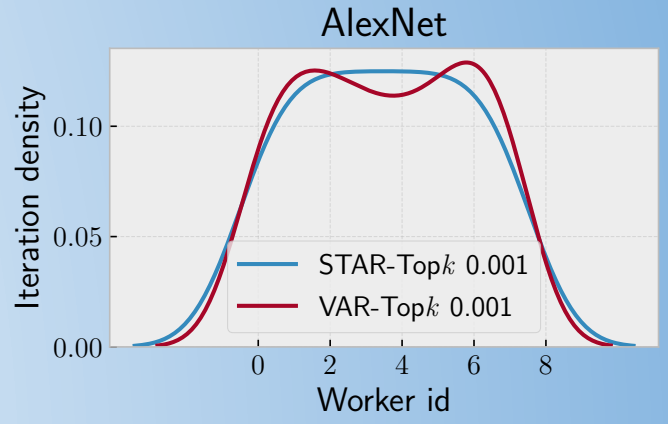








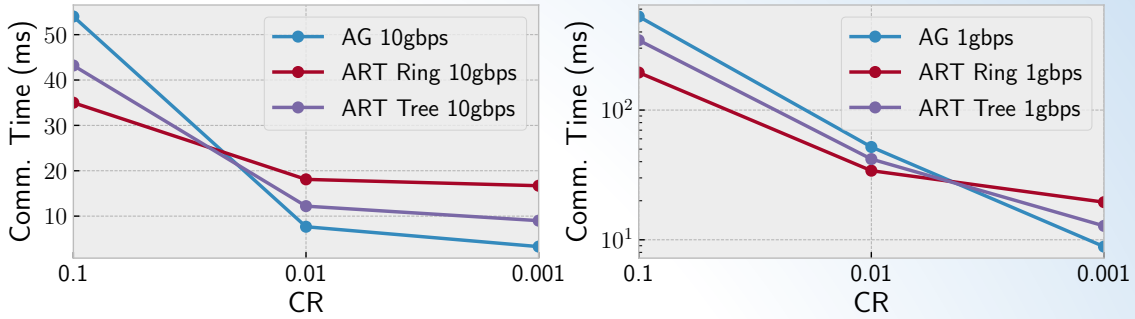




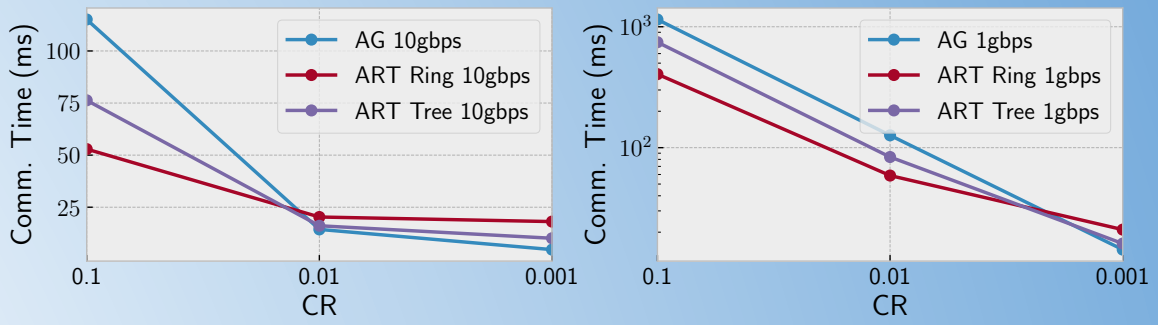
AR-Top*k* with Ring/Tree-AllReduce or AG?

Network latency limited to 1 ms and bandwidth varied to different CRs to compare communication cost

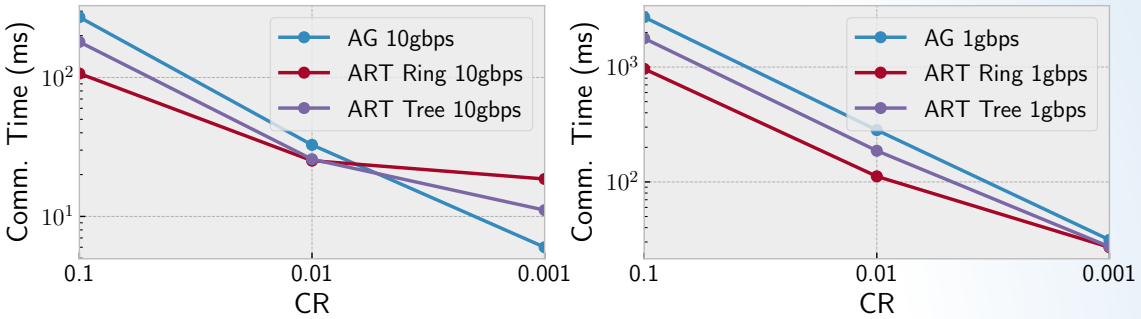
ResNet18



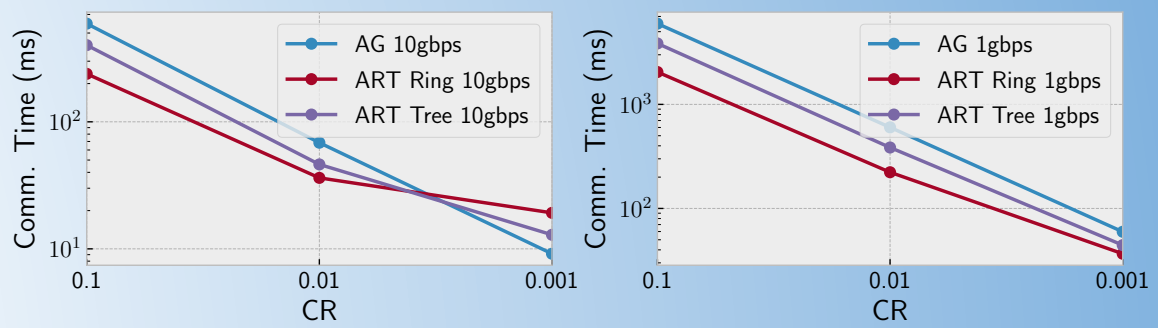
ResNet50



AlexNet



ViT



Communication Cost Analysis

- AR-Topk communication cost varies depending on AllReduce implementation

Comm. Cost of AR-Topk with Ring-AR

$$t^{(c)}_{ARTring} = \alpha[\ 2(N-1)\ +\ \log(N)\ ] + Mc\beta[\ 2\frac{(N-1)}{N}\ +\ \log(N)\ ]$$

Comm. Cost of AR-Topk with Tree-AR

$$t^{(c)}_{ARTtree} = 3\alpha\log(N) + 3Mc\beta\log(N)$$

$$(\frac{\alpha}{\beta})_{\text{ART-Ring\_over\_ART-Tree}} < \left(\frac{\log(N) - (N-1)/N}{N-1-\log(N)}\right)Mc$$

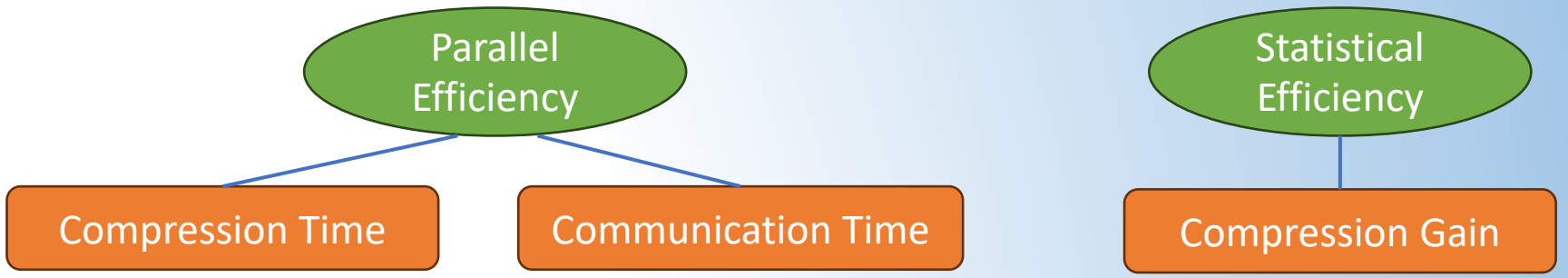
$$(\frac{\alpha}{\beta})_{\text{ART-Ring\_over\_AG}} < \left(1 - \frac{1}{N} - \frac{\log(N)}{2(N-1)}\right)Mc$$

$$(\frac{\alpha}{\beta})_{\text{ART-Tree\_over\_AG}} < \left(\frac{(N-1)}{\log(N)} - \frac{3}{2}\right)Mc$$

Balancing Parallel-Statistical Efficiency in Gradient Compression

- Parallel scaling is influenced by the network, collective used, model-size, cluster-size and target CR
- However, model convergence is also influenced by statistical efficiency of a compressor

Thus, it is crucial to account for both parallel and statistical efficiency in DL compression to attain training speedups while achieving good accuracy



Compression as Multi-Objective Optimization

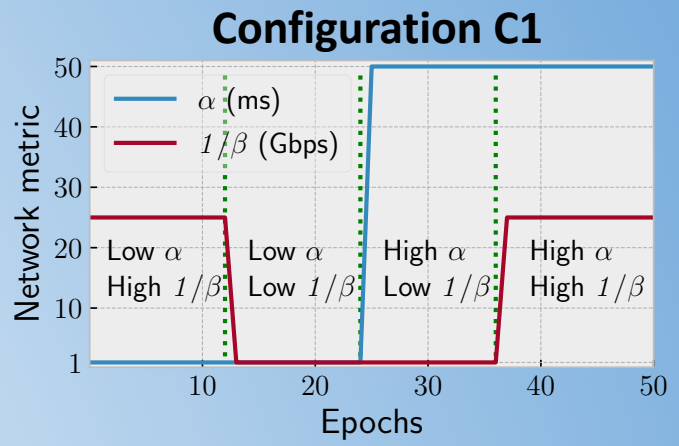
- Parallel and Statistical efficiency are pareto-related!
- Smaller CRs compress more, reducing communication cost; simultaneously degrade model quality as well
- **As gradient sensitivity and network performance can change during training, should we change CR over time as well?**

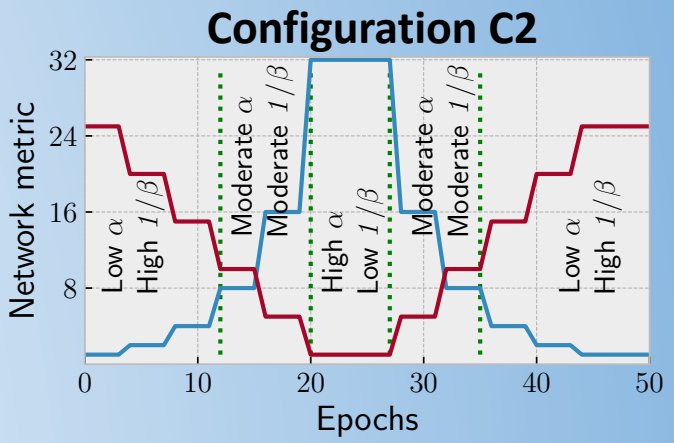
We model gradient compression as a multi-objective optimization problem!

$$c_{optimal} = \underset{c}{\operatorname{argmin}}\{\mathcal{F}(\, t_{comp}^{(c)}, t_{sync}^{(c)}, gain^{-1(c)})\}$$

Compression as Multi-Objective Optimization cont'd

- At any training stage, optimal CR is one that **minimizes** compression and communication time, while **maximizing** gain
- CR is bounded in a min-max exploration space to avoid losing information at low and pay high cost at high CRs
- CR search triggered when latency, bandwidth or gain changes beyond a certain threshold; b/w measured via *iperf* and latency by *traceroute*
- We emulate different network settings by varying latency-bandwidth over each epoch via *tc*





MOO Configuration details

- Set c_low to 0.001 and c_high to 0.1 and CR scaling factor of 3; candidate CRs are thus **0.1**, **0.033**, **0.011**, **0.004** and **0.001**
- Initial exploration phase involves running each CR to measure compress, comm. time and gain
- Trigger exploration when latency, bandwidth or gain changes by 10% or more
- Based on the network and training configuration, optimal communication collective is chosen
- Choose between Ring and Tree-AR in AR-Top*k* uses checkpoint-restore approach over NCCL communication library

MOO-based Adaptive Compression

With this adaptive compression approach, we speedup training while still achieving high accuracy

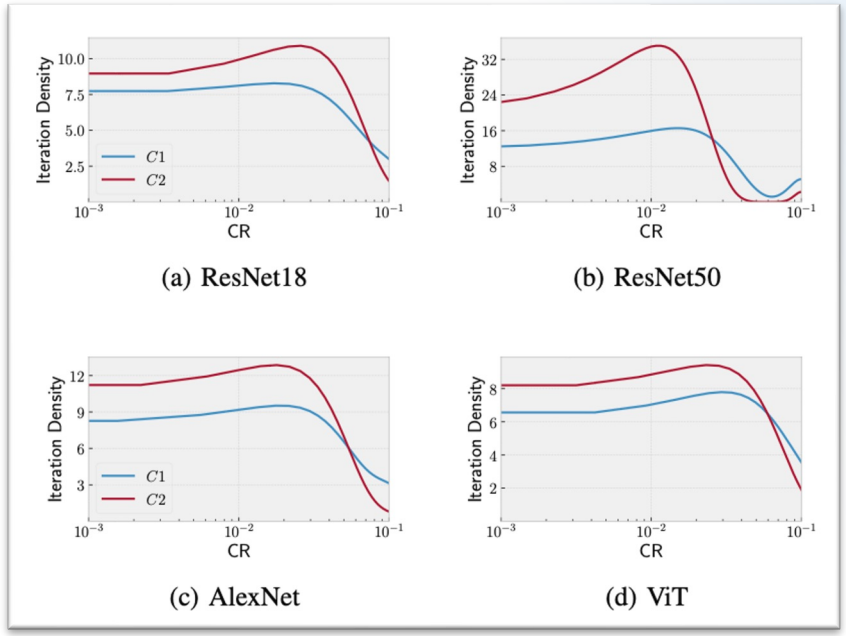
ResNet18: 89.88% (**C1**) and 90.1% (**C2**)

AlexNet: 82.4% (**C1**) and 83.6% (**C2**)

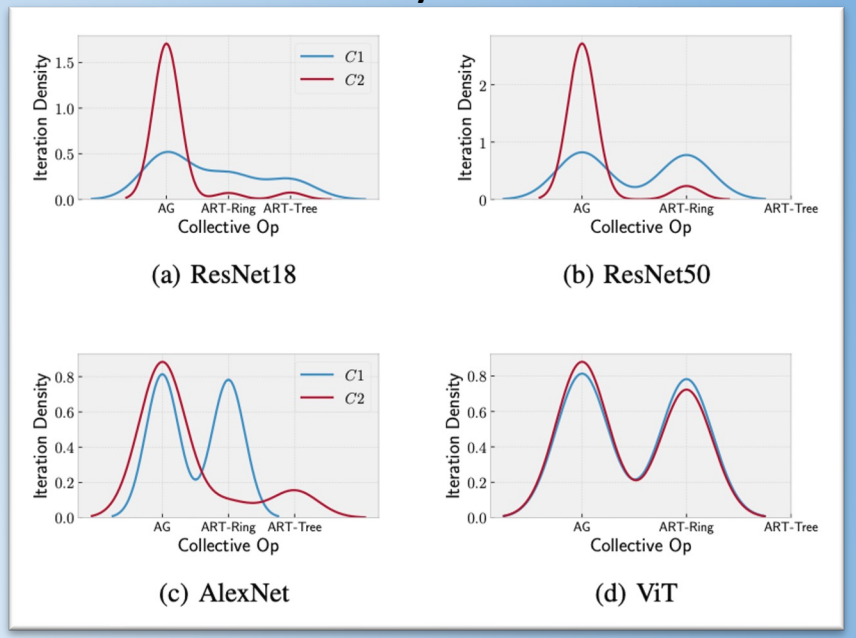
ResNet50: 98.56% (**C1**) and 98.62% (**C2**)

ViT: 80.75% (**C1**) and 81.2% (**C2**)

Distribution density of all candidate CRs



Distribution density of collectives used



Conclusion

- Optimal communication collective depends on network topology, latency, bandwidth, model-size, cluster-size and degree of compression
- AR-Top*k* compression-communication mechanism provides good convergence and uses AR over AG when the former performs better (i.e., has lower communication cost)
- Parallel and statistical efficiency in gradient compression are pareto-related and balanced by modeling compression as a multi-objective optimization problem
- With MOO, we choose CR dynamically and attain convergence quality akin to static compression, and speedup training with an appropriate communication collective

Thank you!

"Flexible Communication for Optimal Distributed Learning over Unpredictable Networks"

https://www.sahiltyagi.com/