



Dual-Way Gradient Sparsification for Asynchronous Distributed Deep Learning

Zijie Yan, Danyang Xiao, MengQiang Chen,
Jieying Zhou, Weigang Wu†

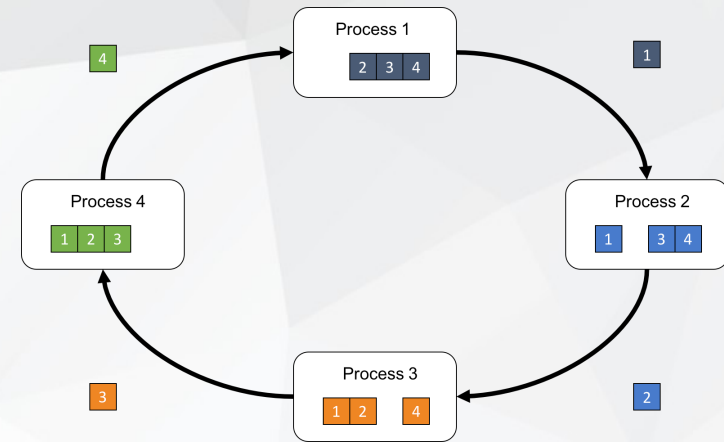
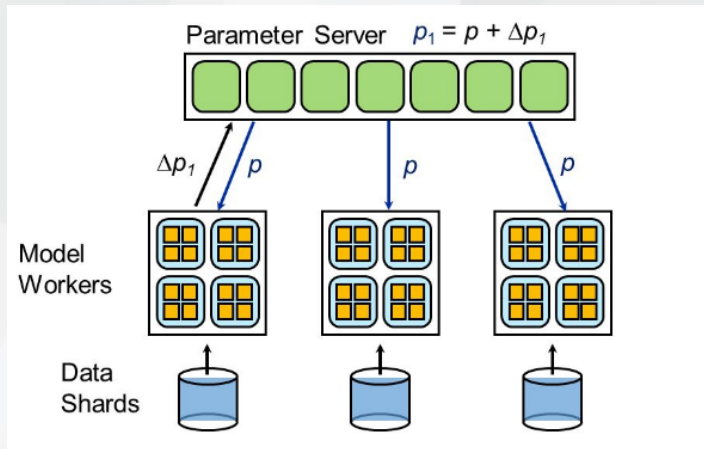
Sun Yat-sen University
Guangzhou, China

Outline



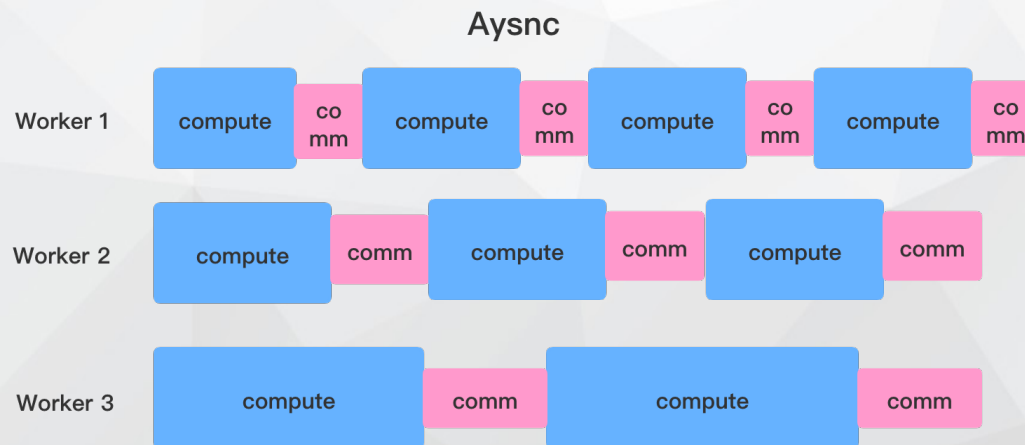
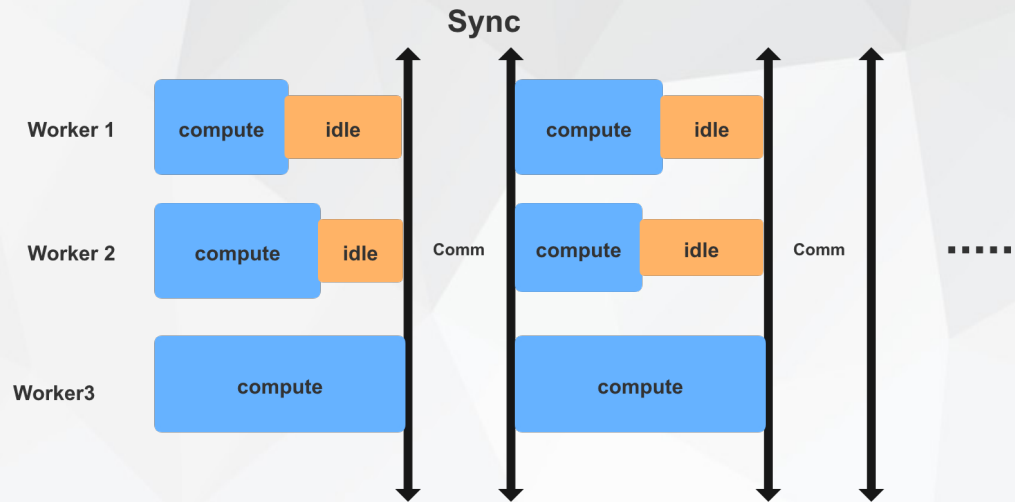
- 1. Introduction**
- 2. The Proposed Algorithm**
- 3. Performance Evaluation**
- 4. Conclusion and Future Work**

Introduction



- Training may take an impractically long time
 - Growing volume of training data (e.g., ImageNet >1TB)
 - More complex model
- Solution: distributed training
 - The common practice of current DL frameworks
 - Enabled by Parameters Servers (PS) or Ring All-Reduce
 - Synchronous SGD or Asynchronous SGD

Introduction

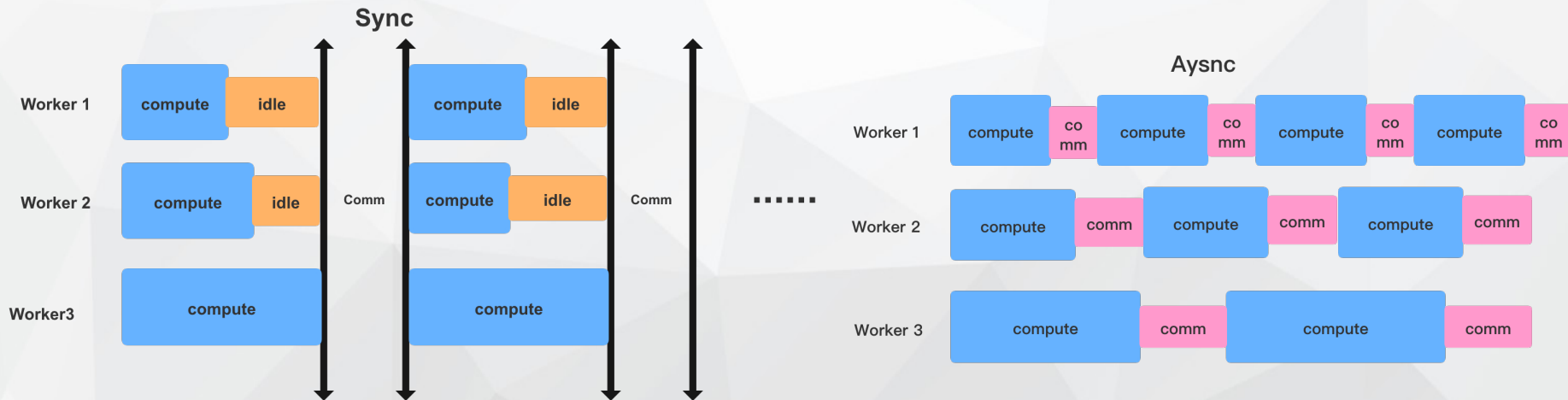


Introduction



Communication overhead: Distributed training can significantly reduce the total computation time. However, the communication overhead seriously affect the efficiency of training.

Solutions: Reduce the frequency / data size of communication.



Introduction

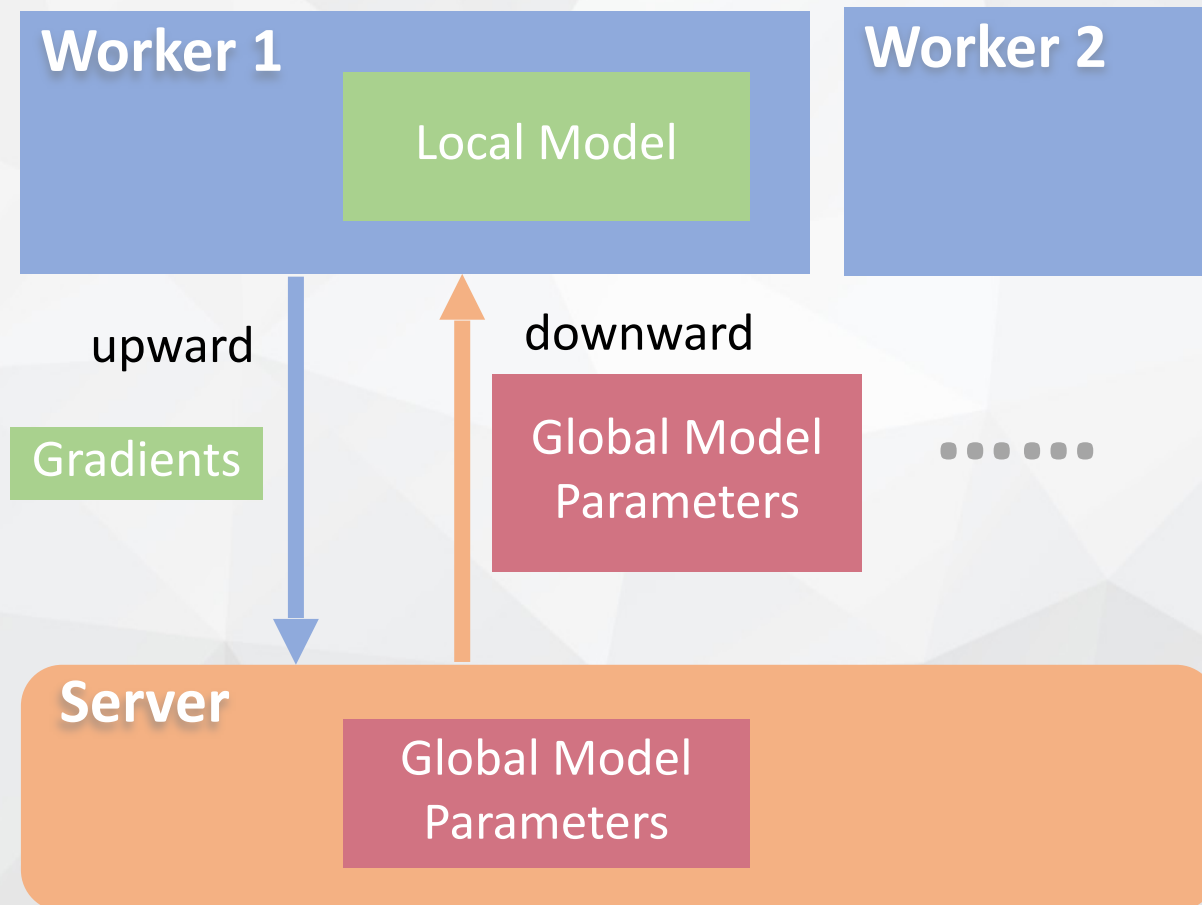


- **Gradient Quantization**
 - **1-Bit SGD, QSGD, TernGrad: use fewer bits to represent value.**
- **Gradient Sparsification**
 - **Threshold Sparsification: only gradients which are greater than a predefined threshold will be sent.**
 - **Gradient Dropping: remove the gradient of R% with the smallest absolute value.**
 - **Deep Gradient Compression: apply momentum correction to correct the disappearance of momentum discounting factor.**

Introduction



PS based ASGD



Contributions

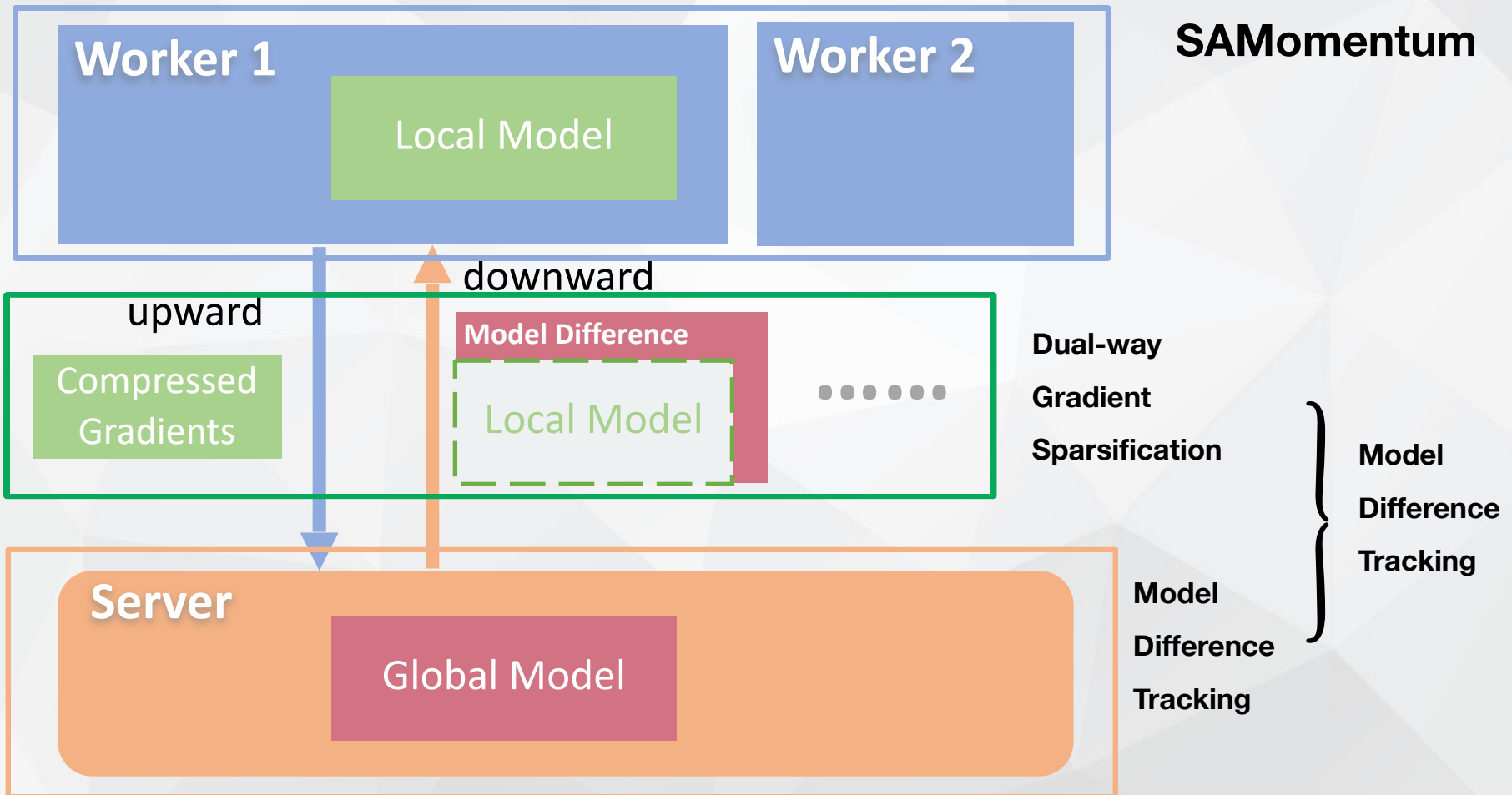


- **Dual-Way Gradient Sparsification (DGS)**
 - **Model Difference Tracking**
 - **Dual-way Gradient Sparsification Operations**
 - **Eliminates the communication bottleneck**
- **Sparsification Aware Momentum (SAMomentum)**
 - **A novel momentum designed for gradient sparsification scenario**
 - **Offers significant optimization**

Contributions



DGS

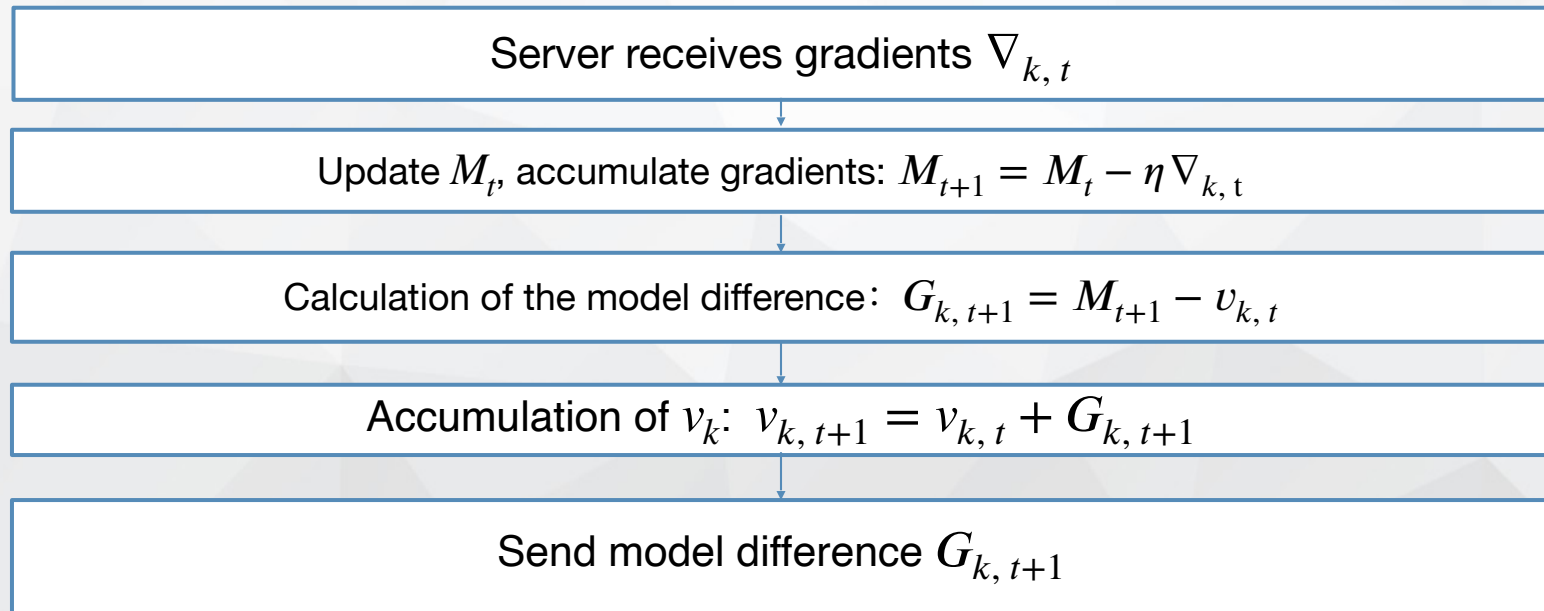


Model Difference Tracking



- Notions

- M_t : The accumulation of updates at the time t .
- $G_{k,t}$: Model difference between the server and the worker k .
- v_k : Accumulation of model difference sent by the server to the worker k .

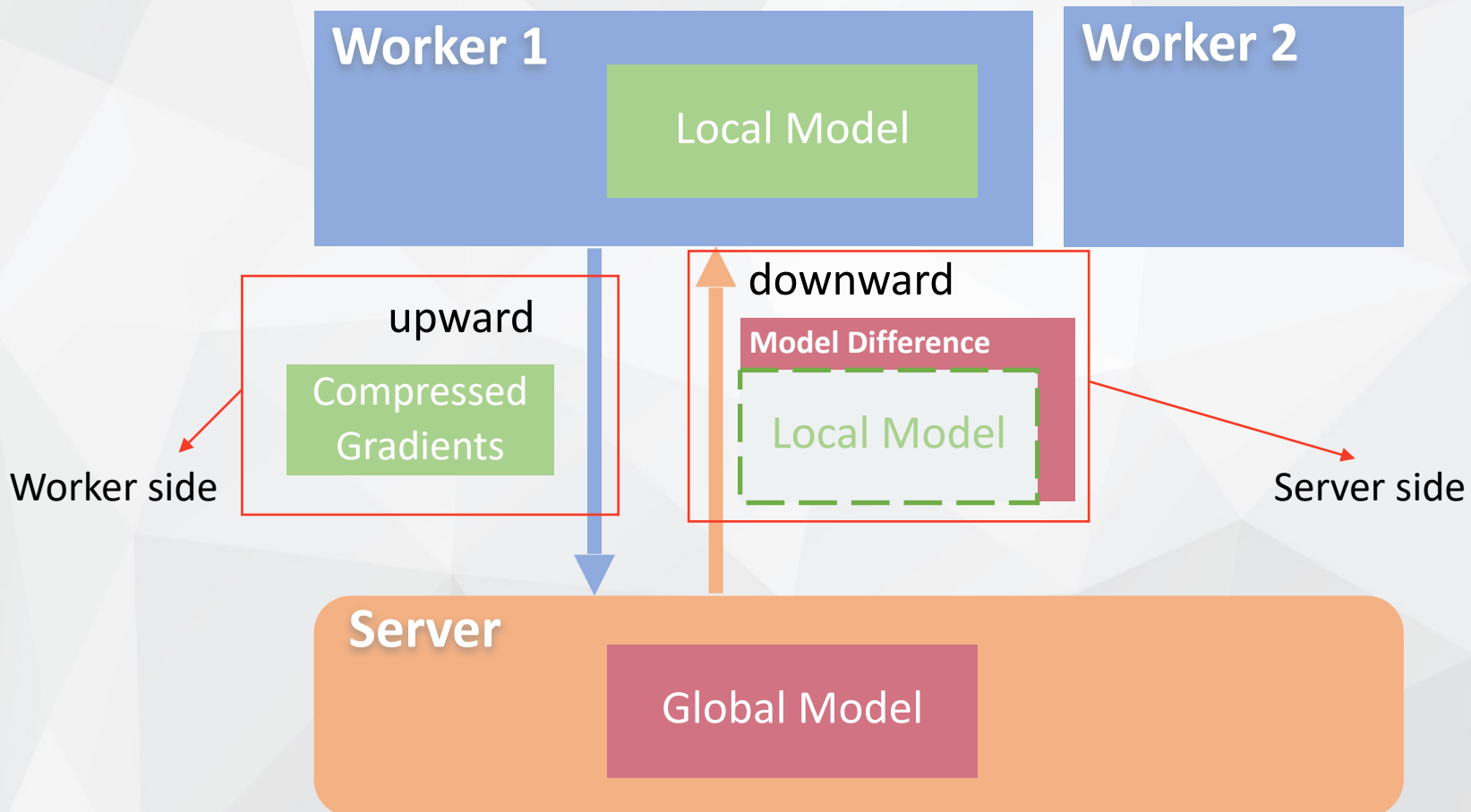


Model Difference Tracking



- What's changed?
 - DGS chooses to transmits model difference $G_{k,t+1}$ rather than the global model
 - Model differences (residual gradients) that have not sent yet are recorded in $M_{t+1} - v_{k,t}$, implicitly avoiding the loss of information.
- Now we can compress the downward communication!

Dual-way Gradient Sparsification



Dual-way Gradient Sparsification - Worker Side



```
3: for  $t = 0, 1, \dots$  do
4:   Sample data  $x$  from  $\mathcal{X}$ 
5:    $\nabla_{k,t} \leftarrow \text{Backward}(x, \theta_{k,\text{prev}(k)})$ 
6:    $r_{k,t} \leftarrow r_{k,\text{prev}(k)} + \eta \nabla_{k,t}$ 
7:   for  $j = 0, \dots, J$  do
8:     // iterate over every layer
9:      $\text{thr} \leftarrow R\% \text{ of } |r_{k,t}[j]|$ 
10:     $\text{Mask} \leftarrow |r_{k,t}[j]| > \text{thr}$ 
11:     $r_{k,t}[j] \leftarrow r_{k,t}[j] \odot \neg \text{Mask}$ 
12:     $g_{k,t}[j] \leftarrow r_{k,t}[j] \odot \text{Mask}$ 
13:  end for
14:  Send  $\text{encode}(g_{k,t})$  to the server
15:  Recieve  $G_{k,t+1}$  from the server
16:   $\theta_{k,t+1} \leftarrow \text{SGD}(\theta_{k,\text{prev}(k)}, \text{decode}(G_{k,t+1}))$ 
```

→ Select threshold

Dual-way Gradient Sparsification - Worker Side



```
3: for  $t = 0, 1, \dots$  do
4:   Sample data  $x$  from  $\mathcal{X}$ 
5:    $\nabla_{k,t} \leftarrow \text{Backward}(x, \theta_{k,\text{prev}(k)})$ 
6:    $r_{k,t} \leftarrow r_{k,\text{prev}(k)} + \eta \nabla_{k,t}$ 
7:   for  $j = 0, \dots, J$  do
8:     // iterate over every layer
9:      $\text{thr} \leftarrow R\% \text{ of } |r_{k,t}[j]|$ 
10:     $\text{Mask} \leftarrow |r_{k,t}[j]| > \text{thr}$ 
11:     $r_{k,t}[j] \leftarrow r_{k,t}[j] \odot \neg \text{Mask}$ 
12:     $g_{k,t}[j] \leftarrow r_{k,t}[j] \odot \text{Mask}$ 
13:  end for
14:  Send  $\text{encode}(g_{k,t})$  to the server
15:  Recieve  $G_{k,t+1}$  from the server
16:   $\theta_{k,t+1} \leftarrow \text{SGD}(\theta_{k,\text{prev}(k)}, \text{decode}(G_{k,t+1}))$ 
```

→ Sparsification

Dual-way Gradient Sparsification - Server Side



Model Difference Tracking

```
2: while Receive  $encode(g_{k,t})$  from worker  $k$  do  
3:    $M_{k,t+1} \leftarrow M_{k,t} - g_{k,t}$   
4:    $G_{k,t+1} \leftarrow M_{t+1} - v_{k,prev(k)}$   
5:   if Need secondary compression then  
6:     for  $j = 0, \dots, J$  do  
7:       // iterate over every layer  
8:        $thr \leftarrow R\%$  of  $|G_{k,t+1}[j]|$   
9:        $Mask \leftarrow |G_{k,t+1}[j]| > thr$   
10:       $G_{k,t+1}[j] \leftarrow G_{k,t+1}[j] \odot Mask$   
11:    end for  
12:  end if  
13:  Send  $encode(G_{k,t+1})$  to the worker  $k$   
14:   $v_{k,t+1} \leftarrow v_{k,prev(k)} - G_{k,t+1}$   
15:   $prev(k) \leftarrow t + 1$ 
```

Dual-way Gradient Sparsification - Server Side



Secondary compression

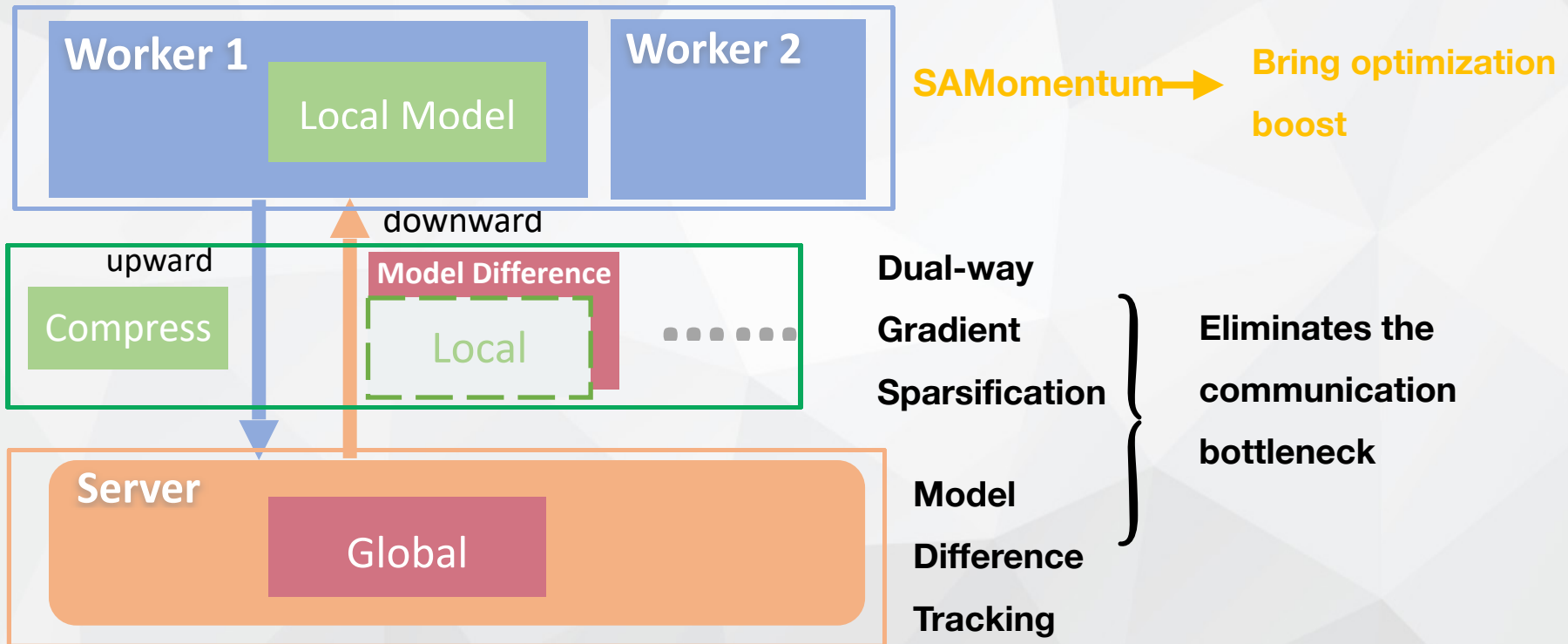
- Secondary compression guarantees the sparsity of the send-ready model difference in downward communication, no matter how many workers are running.
- The server implicitly accumulates remaining gradient locally.
- Eliminates the overhead of the downward communication.

```
2: while Receive  $encode(g_{k,t})$  from worker  $k$  do
3:    $M_{k,t+1} \leftarrow M_{k,t} - g_{k,t}$ 
4:    $G_{k,t+1} \leftarrow M_{t+1} - v_{k,prev(k)}$ 
5:   if Need secondary compression then
6:     for  $j = 0, \dots, J$  do
7:       // iterate over every layer
8:        $thr \leftarrow R\%$  of  $|G_{k,t+1}[j]|$ 
9:        $Mask \leftarrow |G_{k,t+1}[j]| > thr$ 
10:       $G_{k,t+1}[j] \leftarrow G_{k,t+1}[j] \odot Mask$ 
11:     end for
12:   end if
13:   Send  $encode(G_{k,t+1})$  to the worker  $k$ 
14:    $v_{k,t+1} \leftarrow v_{k,prev(k)} - G_{k,t+1}$ 
15:    $prev(k) \leftarrow t + 1$ 
```


Sparsification Aware Momentum



DGS



SAMomentum - Background



- **Momentum** is commonly used in deep training, which is known to offer a significant optimization boost.
- However, indeterminate update intervals in gradient sparsification will result in the **disappearance of momentum**.

SAMomentum - Background



Dense update :

$$u_t = mu_{t-1} + \eta \nabla_t, \theta_{t+1} = \theta_t - u_t$$

After T updates

$$\text{Dense } u_{t+T}^{(i)} = \eta \left[\dots + m^{T-2} \nabla_{t+2}^{(i)} + m^{T-1} \nabla_{t+1}^{(i)} \right] + m^T u_t^{(i)}$$

$u_t^{(i)}$ denotes the i -th position of a flattened velocity u_t

SAMomentum - Background



Sparse update :

$$r_{k,t} = r_{k,t-1} + \eta * \nabla_{k,t}, \quad u_t = mu_{t-1} + \text{sparsify}(r_{k,t})$$

$$r_{k,t} = \text{unsparsify}(r_{k,t}), \quad \theta_{t+1} = \theta_t - u_t$$

Remaining Gradients

After T updates

$$\text{Sparse } u_{t+T}^{(i)} = \eta \left[\dots + \nabla_{t+2}^{(i)} + \nabla_{t+1}^{(i)} \right] + m^T u_t^{(i)}$$

$u_t^{(i)}$ denotes the i -th position of a flattened velocity u_t

Momentum Disappearing



$$\text{Dense } u_{t+T}^{(i)} = \eta \left[\dots + m^{T-2} \nabla_{t+2}^{(i)} + m^{T-1} \nabla_{t+1}^{(i)} \right] + m^T u_t^{(i)}$$

$$\text{Sparse } u_{t+T}^{(i)} = \eta \left[\dots + \nabla_{t+2}^{(i)} + \nabla_{t+1}^{(i)} \right] + m^T u_t^{(i)}$$

- Momentum factor m controls the proportion of historical information.
- The disappearance of m impairs the convergence performance.

SAMomentum



$$u_{k,t} = mu_{k,\text{prev}(k)} + \eta \nabla_{k,t} + \text{unsparsify} \left(mu_{k,\text{prev}(k)} + \eta \nabla_{k,t} \right) * \left(\frac{1}{m} - 1 \right)$$

$$g_{k,t} = \text{sparsify} \left(mu_{k,\text{prev}(k)} + \eta \nabla_{k,t} \right)$$

$$\theta_{t+1} = \theta_t - g_{k,t}$$

SAMomentum



From parameter perspective:

$$u_{k,c}^{(i)} = \begin{cases} mu_{k,c-1}^{(i)} + \eta \nabla_{k,c}^{(i)} & > thr \\ \left(mu_{k,c-1}^{(i)} + \eta \nabla_{k,c}^{(i)} \right) * \frac{1}{m} & \leq thr \end{cases}$$

↓ Send $u_k^{(i)}$ at c and $c + T$

$$\begin{aligned} u_{k,c+T}^{(i)} &= mu_{k,c+T-1}^{(i)} + \eta \nabla_{k,c+T}^{(i)} \\ &= m \left(\left(mu_{k,c+T-2}^{(i)} + \eta \nabla_{k,c+T-1}^{(i)} \right) * \frac{1}{m} \right) + \eta \nabla_{k,c+T}^{(i)} \\ &= mu_{k,c+T-2}^{(i)} + \eta \nabla_{k,c+T-1}^{(i)} + \eta \nabla_{k,c+T}^{(i)} \\ &= \dots \\ &= mu_{k,c}^{(i)} + \eta \sum_{i=1}^T \nabla_{k,c+i}^{(i)} \end{aligned}$$

$u_t^{(i)}$ denotes the i -th position of a flattened velocity u_t

SAMomentum and Enlarged Batch Size



SAMomentum

$$\begin{aligned}u_{k,c+T}^{(i)} &= mu_{k,c+T-1}^{(i)} + \eta \nabla_{k,c+T}^{(i)} \\&= m \left(\left(mu_{k,c+T-2}^{(i)} + \eta \nabla_{k,c+T-1}^{(i)} \right) * \frac{1}{m} \right) + \eta \nabla_{k,c+T}^{(i)} \\&= mu_{k,c+T-2}^{(i)} + \eta \nabla_{k,c+T-1}^{(i)} + \eta \nabla_{k,c+T}^{(i)} \\&= \dots \\&= mu_{k,c}^{(i)} + \eta \sum_{i=1}^T \nabla_{k,c+i}^{(i)}\end{aligned}$$

Enlarged Batch Size

$$\begin{aligned}u_{k,c+T}^{(i)} &= mu_{k,c}^{(i)} + T\eta * \frac{1}{T} \left(\nabla_{k,c+1}^{(i)} + \dots + \nabla_{k,c+T}^{(i)} \right) \\&= mu_{k,c}^{(i)} + \eta \sum_{i=1}^T \nabla_{k,c+i}^{(i)}\end{aligned}$$

Experiments Setup



1. Comparison to Other Algorithms

- **Dense:**
 - Single node momentum SGD
 - Asynchronous SGD
- **Sparse:**
 - Gradient Dropping (EMNLP 2017)
 - Deep Gradient Compression (ICLR 2018, STOA)

2. Datasets

- **ImageNet**
- **CIFAR-10**

Scalability and Generalization Ability

CIFAR-10



Workers in total	Batchsize per worker	Training Method	Top-1 Accuracy
1	256	MSGD	93.08% -
		ASGD	91.54% -1.54%
		GD-async	92.15% -0.93%
		DGC-async	92.75% -0.33%
		DGS	92.97% -0.11%
4	128	ASGD	90.7% -2.38%
		GD-async	92.01% -1.07%
		DGC-async	92.64% -0.44%
		DGS	92.91% -0.17%
8	64	ASGD	90.46% -2.62%
		GD-async	91.81% -1.27%
		DGC-async	92.37% -0.71%
		DGS	93.32% +0.24%
16	32	ASGD	90.53% -3.01%
		GD-async	91.43% -1.65%
		DGC-async	92.28% -0.80%
		DGS	92.98% -0.10%
32	16	ASGD	88.36% -4.71%
		GD-async	91% -2.08%
		DGC-async	91.86% -1.22%
		DGS	92.69% -0.39%

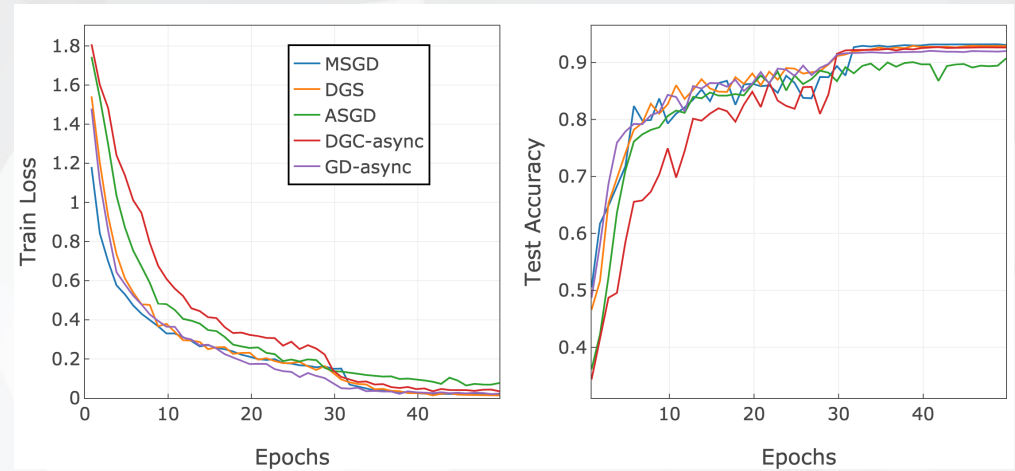


Fig. 4 nodes

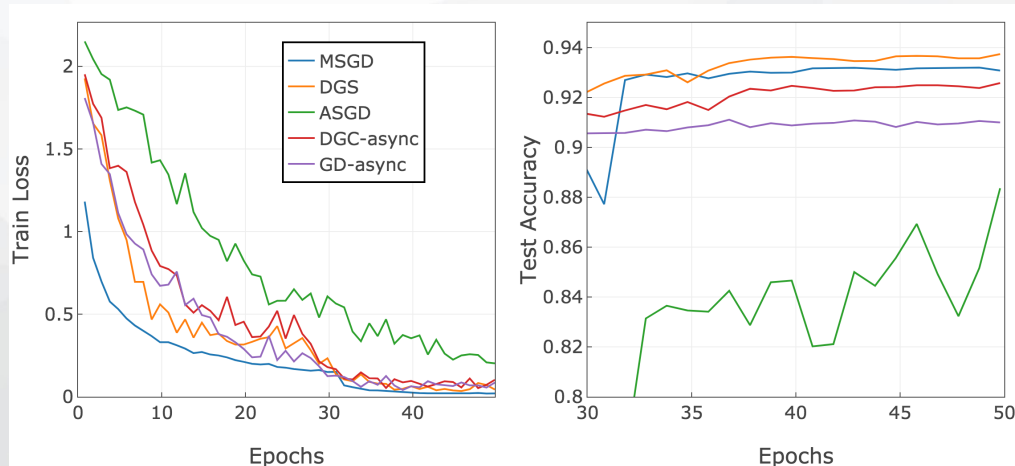


Fig. 32 nodes

Scalability and Generalization Ability

ImageNet



Workers in total	Batchsize per iteration	Training Method	Top-1 Accuracy
1	256	MSGD	69.40% -
4		ASGD	66.68% -2.72%
		GD-async	66.26% -3.14%
		DGC-async	68.37% -1.03%
		DGS	69.00% -0.40%
16		ASGD	66.25% -3.15%
		GD-async	66.19% -3.21%
		DGC-async	67.62% -1.78%
		DGS	68.25% -1.15%

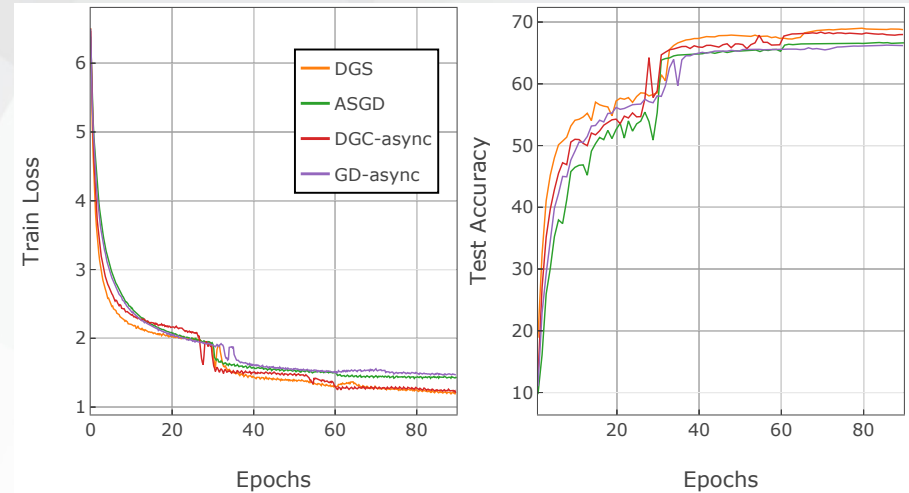


Fig. 4 nodes

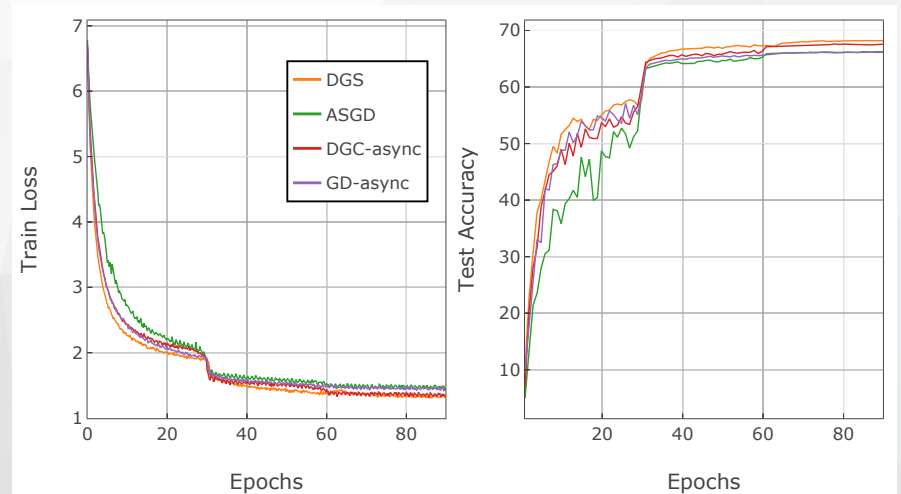


Fig. 16 nodes

Low Bandwidth Results

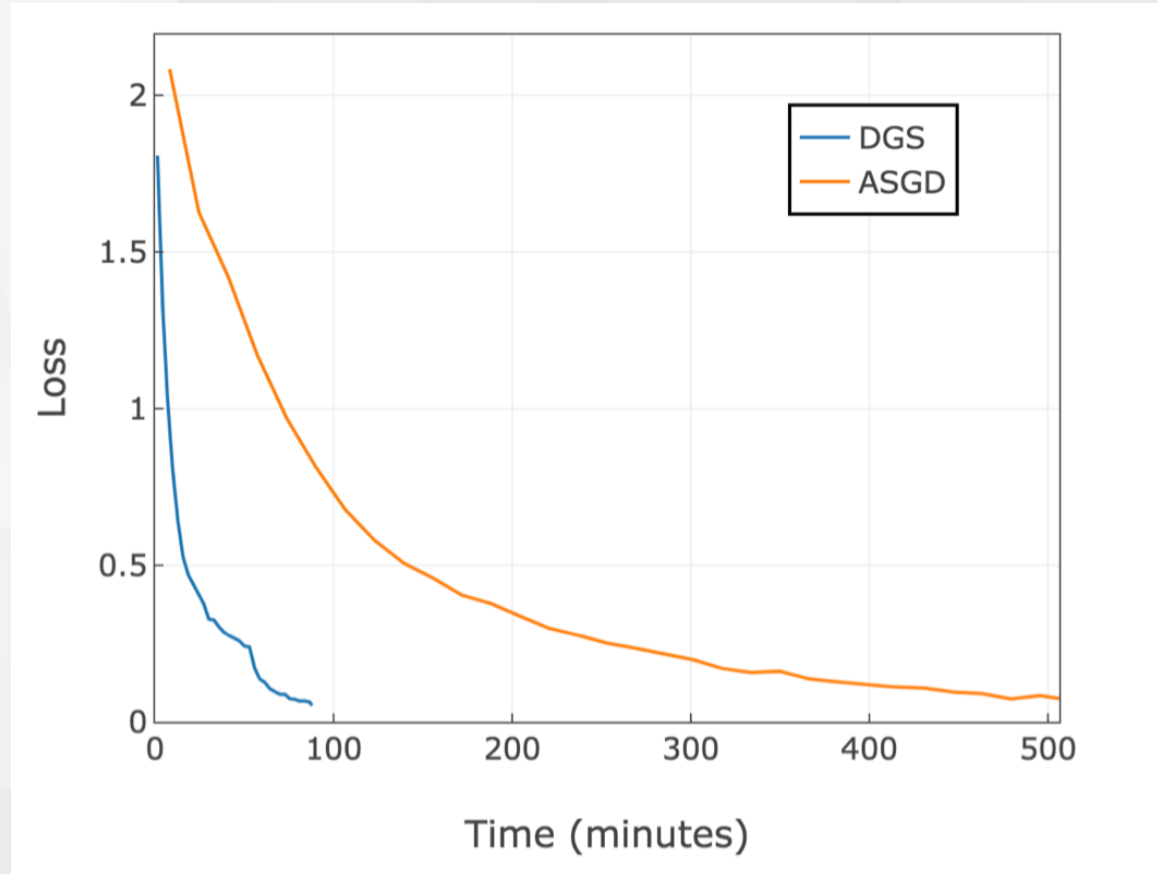


Fig : Time vs Training Loss on 8 workers with 1Gbps Ethernet

Speed up

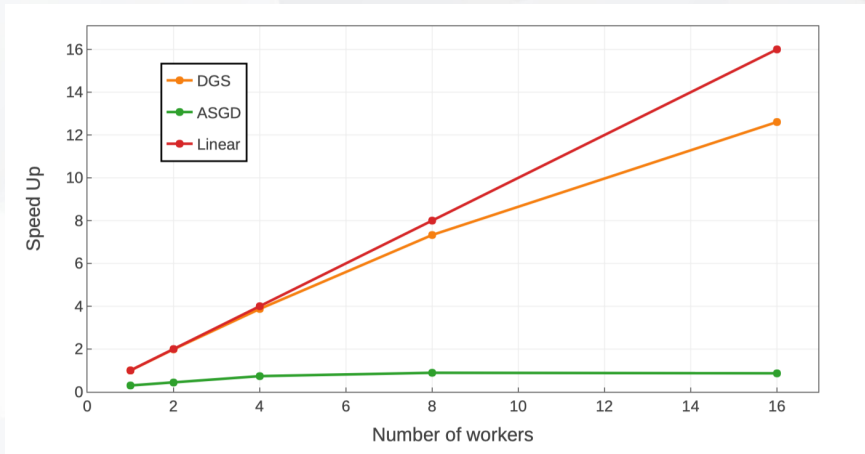
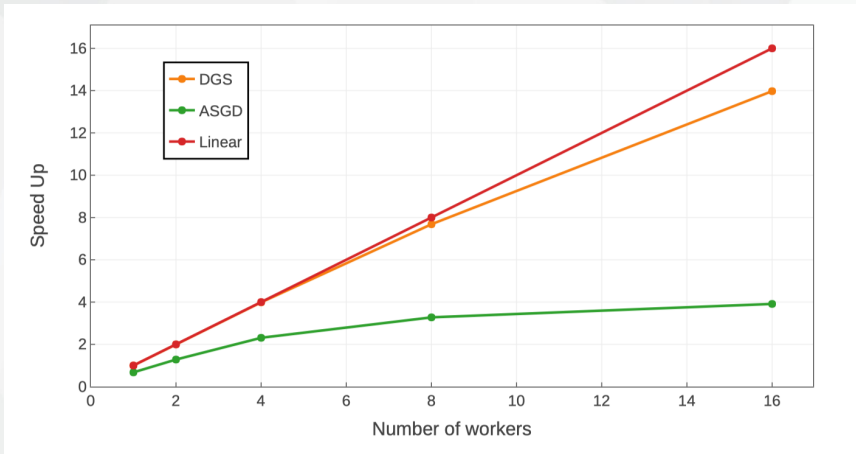


Fig : Speedups for DGS and ASGD on ImageNet with 10Gbps and 1Gbps Ethernet

Conclusion and Future Work



Conclusion

1. Enable dual-way sparsification for PS-based asynchronous training.
2. Introduce SAMomentum to bring significant optimization.
3. Experiment results show that DGS outperforms existing routing algorithms.

Future Work

1. Apply SAMomentum to synchronous training.
2. Combine DGS with other compression approaches.



Thanks for listening