Scalable Machine Learning and Deep Learning - Review Questions 6

By

Akhil Yerrapragada (akhily@kth.se)

Gibson Chikafa (chikafa@kth.se)

Group Name – gibson_akhil

1. **1 point.** Explain how the data-parallelized learning work?

Ans: Let us consider we have 4 workers. In data parallelization, we replicate the whole model on every worker and train the model using dataset available for that worker. Each worker computes the loss function using the dataset available for it and computes the gradient for the loss function. The computed local gradients from workers are then sent to the central server and the central server performs the model aggregation using an aggregation function that computes the average of the received gradients. The central server shares the weights with all the workers which will be used for next iteration.

---

2. **1 point.** Explain different synchronization approaches in data-parallelized learning?

Ans:

There are four synchronization approaches in data-parallelized learning:

1) Synchronous:

Each worker does feedforward, backward propagation and sends the gradients to central server. Here all the workers must wait for other workers to finish sending the gradients in current iteration to central server. The central server post receiving all gradients performs the aggregation and updates the weights to each worker and the second iteration continues. This approach can cause straggler problem as one delayed worker can affect overall throughput.

2) Stale-Synchronous:

The stale-synchronous approach avoids straggler problem. In this approach we let fast workers to do more updates than slower workers. To limit the gap between fast and slow worker, a staleness bound barrier is defined.

$$\mathbf{w}_{i,t+1} := \mathbf{w}_0 - \eta\left(\sum_{k=1}^{t}\sum_{j=1}^{n} G_{j,k} + \sum_{k=t-s}^{t} G_{i,k} + \sum_{(j,k)\in S_{i,t+1}} G_{j,k}\right)$$

The first term here defines the gradients till the staleness barrier, the second term is gradients sent by worker I in the barrier. The last is gradients sent by other workers in between the barrier.

3) Asynchronous:

Aggregations are computed at central server for every t seconds. Here, there is no guarantee that the gradients from all workers will be received in the specified time. Accuracy achieved using this approach can be less since all aggregations are not considered.

$$\mathbf{w}_{t+1} := \mathbf{w}_t - \eta\sum_{i=1}^{n} G_{i,t-\tau_{k,i}}$$

4) Local SGD:

Each worker completes multiple iterations locally (local computation). After t seconds aggregations will be computed for gradients and the computed average will be used to update the weights.

$$\mathbf{w}_{i,t+1} = \begin{cases} \mathbf{w}_{i,t} - \eta G_{i,t} & \text{if } t+1 \notin \mathcal{I}_T \\ \mathbf{w}_{i,t} - \eta\frac{1}{n}\sum_{i=1}^{n} G_{i,t} & \text{if } t+1 \in \mathcal{I}_T \end{cases}$$

3. **1 point.** Briefly explain gradient quantization and gradient sparsification.

Ans:
Both are communication compression techniques used to reduce the network traffic

1) Gradient Quantization:

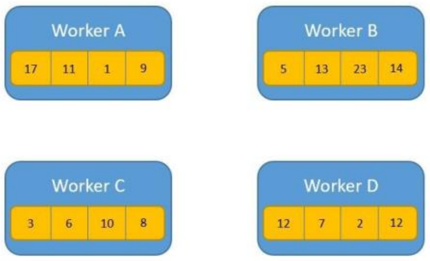
Instead of sending all bits that represent a gradient to the parameter server, only some bits that represents the gradient are shared. Though this approach reduces accuracy, it reduces network traffic significantly.

2) Gradient Sparsification:

Instead of sending all the gradients to parameter server, only gradients that are significant for aggregate computation will be shared. For example, gradients that are zero doesn't need to be shared since they do not contribute to aggregation.

---

4. **1 point.** Use the following picture and show step-by-step how the ring-allreduce works to compute the maximum of all elements?

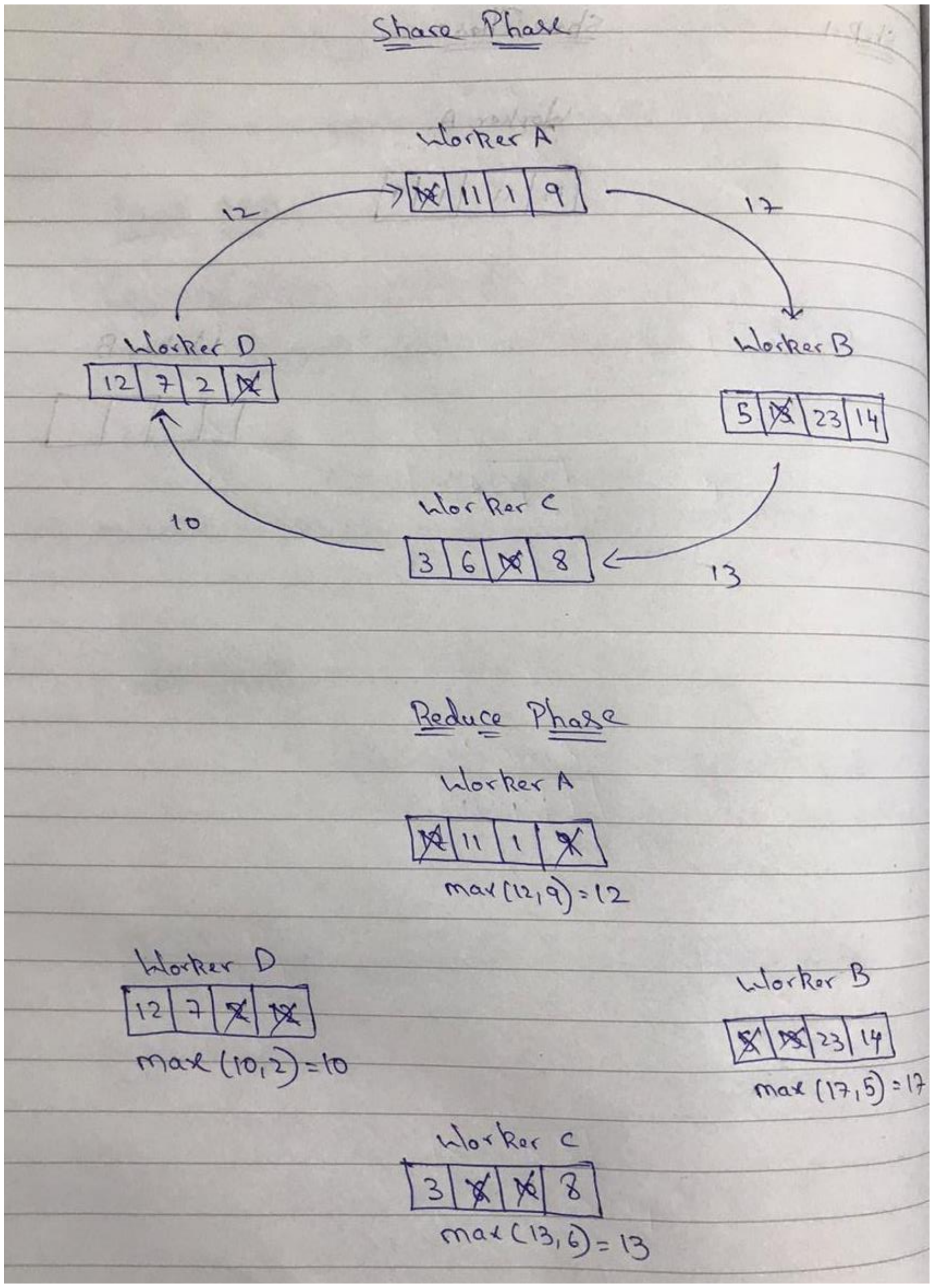


Ans:

Step 1:

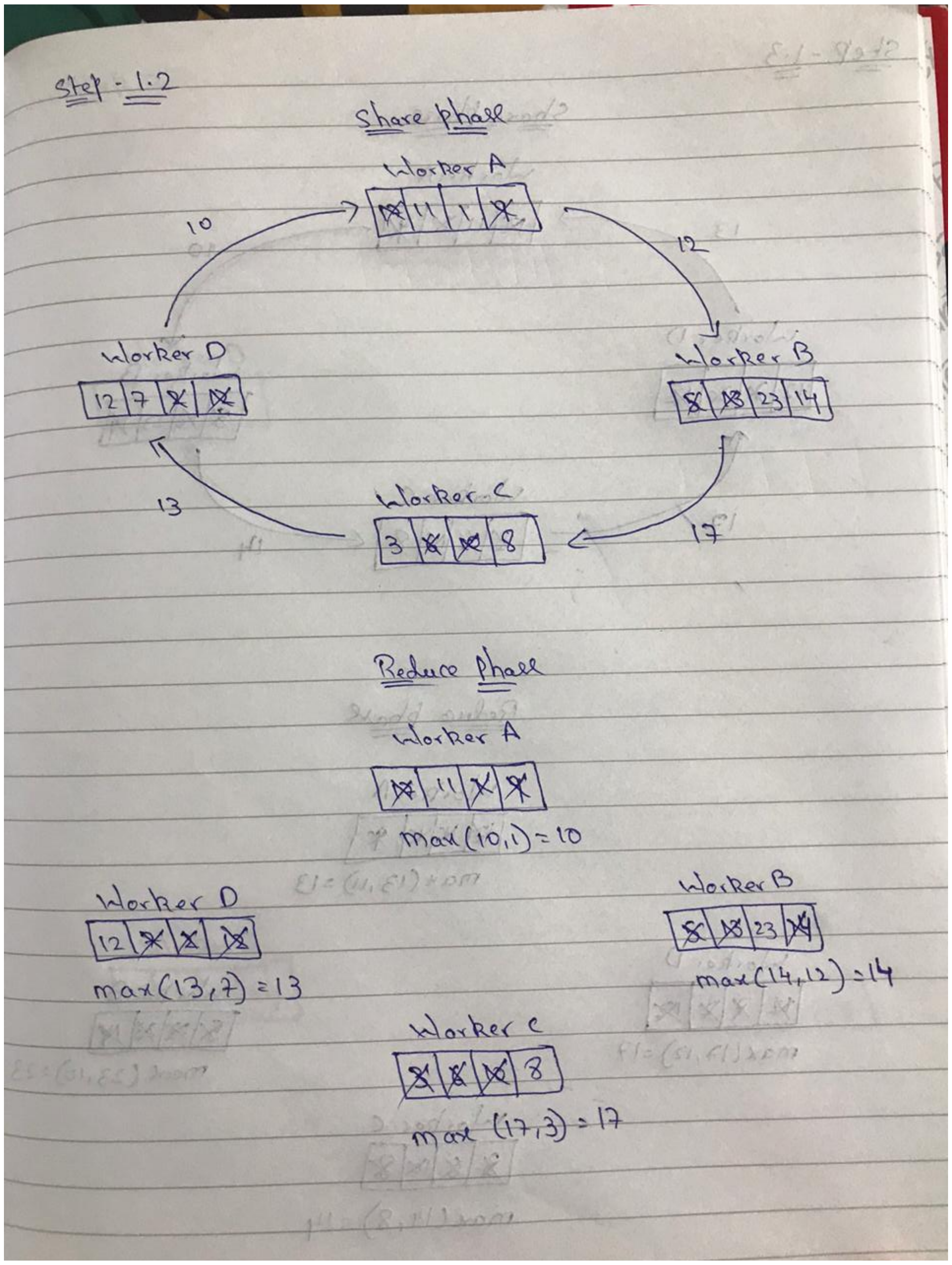
Share – Reduce phase: We repeat share-reduce phase p-1 times. Where, p is 4 since we have four processes/workers. Therefore, we have 3 rounds, and the implementation is as depicted below for each round.

Round 1:

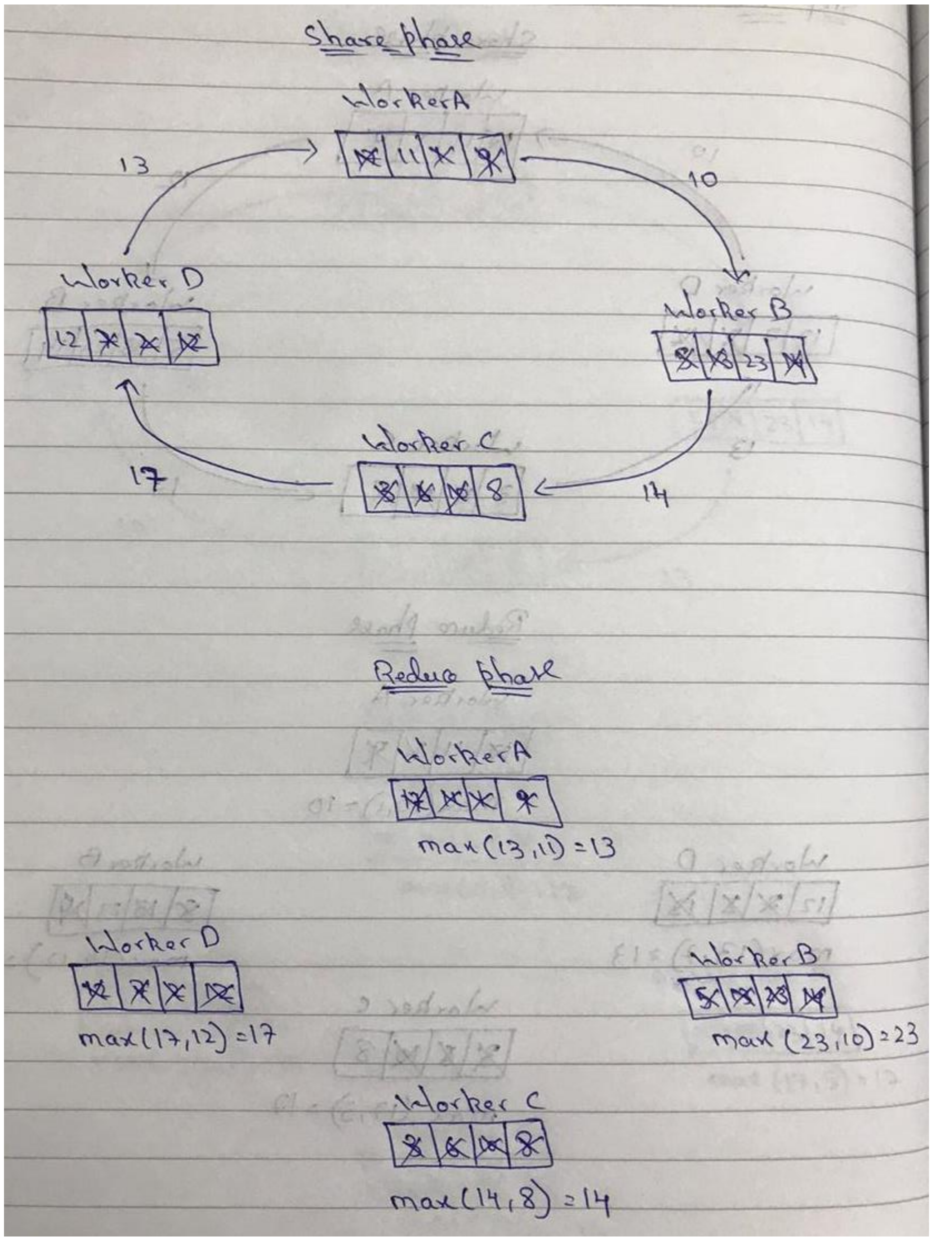


Share Phase
Worker A
12
11
1
9
17
12
Worker B
5
23
14
13
Worker C
3
6
8
10
Worker D
12
7
2
Reduce Phase
Worker A
11
1
max(12,9)=12
Worker D
12
7
max(10,2)=10
Worker B
23
14
max(17,5)=17
Worker C
3
8
max(13,6)=13

Round 2:


Step - 1.2
Share phase
Worker A
11 1
10
12
Worker D
12 7
Worker B
23 14
13
Worker C
3 8
17
Reduce Phase
Worker A
11
max(10,1) = 10
Worker D
12
max(13,7) = 13
Worker B
23
max(14,12) = 14
Worker C
8
max (17,3) = 17

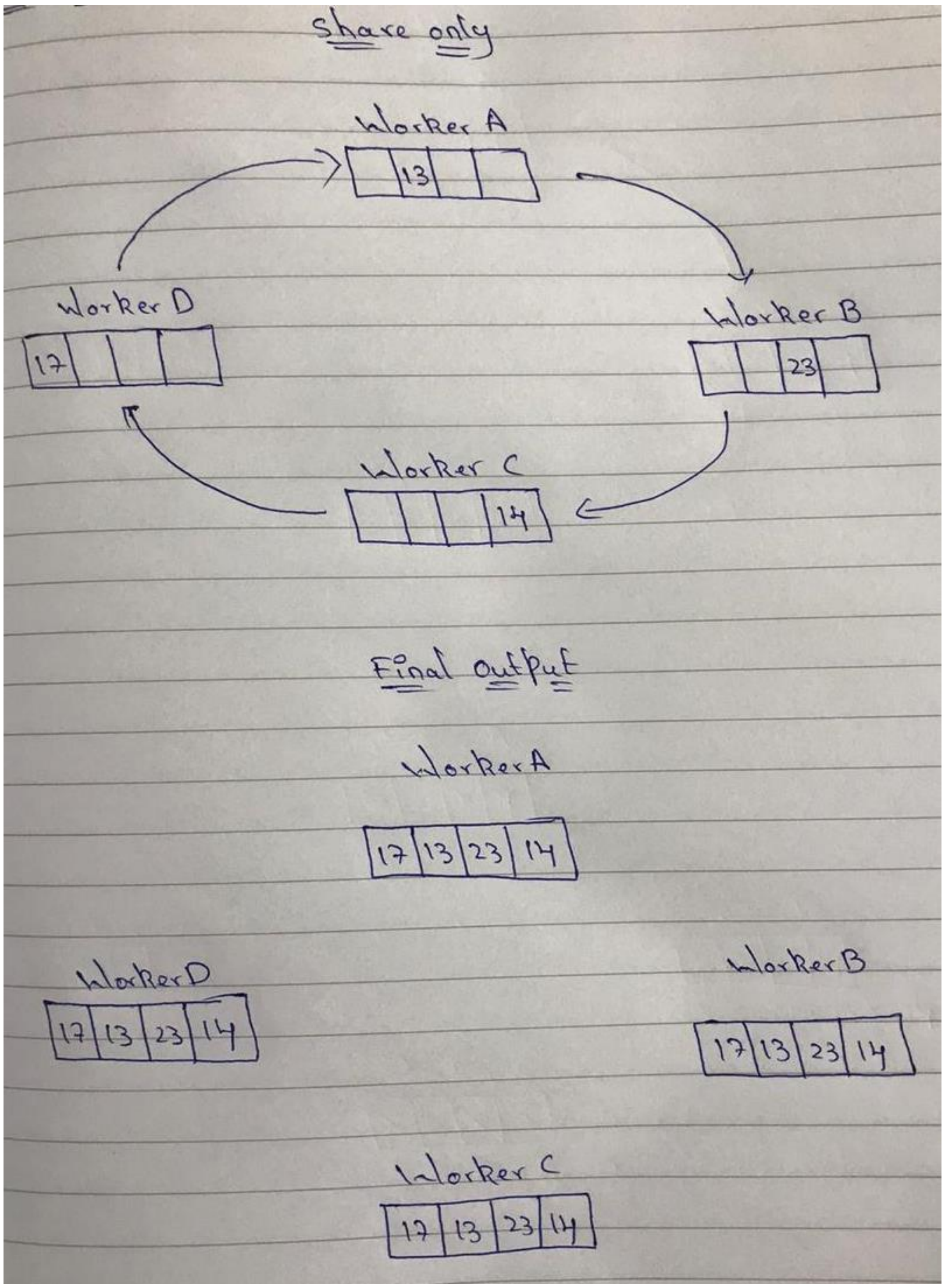
Round 3:



Share phase
Worker A
11
10
Worker B
23
14
Worker C
8
17
Worker D
12
13
Reduce phase
Worker A
max(13,11) = 13
Worker B
max(23,10) = 23
Worker C
max(14,8) = 14
Worker D
max(17,12) = 17

Step 2:

Share only phase where we only share the max values to all the processes.



5. **1 point.** How does the batch-size affect the performance of data-parallelized learning?

Ans: Bigger batch-size can improve the training time performance in data-parallelized learning since smaller batch-sizes underutilize GPUs. However, there are some studies that indicate that bigger batch sizes can lead to poor accuracy. In such cases where accuracy is concerned, small batch sizes can be used.

CROSSBOW address this problem of achieving faster training times with small batch sizes. It does so by replicating the model inside a GPU and train them parallelly. Synchronization among the replicas inside a GPU can be achieved by using a approach similar to Local SGD where each worker computes gradients locally and aggregates after some iterations to obtain average model.