Approximate Gradient Synchronization with AQGB

Xue Liu Shouxi Luo Ke Li Huanlai Xing
Southwest Jiaotong University
Chengdu, China

CCS CONCEPTS

• **Networks → Network protocols**.

KEYWORDS

Machine learning, distributed training, adaptive quantization

ACM Reference Format:
Xue Liu Shouxi Luo Ke Li Huanlai Xing. 2022. Approximate Gradient Synchronization with AQGB. In *6th Asia-Pacific Workshop on Networking (APNet 2022), July 1–2, 2022, Fuzhou, China.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3542637.3543708

1 INTRODUCTION

Recent studies have shown that, in large-scale distributed deep learning, the communication triggered by model synchronization could dominate the entire training, becoming the system bottleneck. Such a problem is getting more serious as large models are getting popular. To deal with this, numerous optimization designs including data compression and pipelining are proposed, to maximize the overlap between the involved communication and computation [1, 3]. Among them, gradient quantization is promising and widely employed—by transmitting quantized-yet-error-compensated gradients rather than original values it can reduce the involved traffic load significantly [2]. However, as a lossy compression technique, gradient quantization has the possible cost of reduced model accuracy or increased rounds to convergence [3].

In large-scale shared clusters, because various distributed applications are likely to coexist, the available bandwidth a transfer could use is time-varying. Accordingly, the perfect quantization scheme should have the ability to adapt its level of compression respecting the network dynamically. Unfortunately, to the best of our knowledge, none of the existing schemes have supported this, as they all adopt fixed and inflexible quantization designs [3].

To fill the gap, we propose the novel design of network-aware adaptive gradient quantization, based on which, we further design AQGB (Adaptive Quantized Gradient Broadcast) to achieve approximate gradient synchronization for distributed training efficiently. Although attractive, making the above idea come true is quite challenging and we adopt three novel designs to address this:

- A flexible and efficient encode/decode scheme suite that supports hierarchical gradient quantization;
- A network-aware quantization ratio control algorithm that dynamically adjusts the level of quantization/compression respecting the network state;
- AQGB, an approximate-yet-consistent gradient synchronization scheme optimizing the overlap between communications and computations for peer-to-peer distributed training.

Corresponding author: Shouxi Luo (sxluo@swjtu.edu.cn). This work was partially supported by NSFC Project 62002300, NSFSC Project 2022NSFSC0944, and Project of Network and Data Security Key Laboratory in Sichuan Province NDS2022-1.

APNet 2022, July 1–2, 2022, Fuzhou, China

ACM ISBN 978-1-4503-9748-3/22/07…$15.00
https://doi.org/10.1145/3542637.3543708

2 AQGB

In deep learning, the computation involved in each round of training can be divided into two phases, namely *forward propagation* (FP) and *backward propagation* (BP). Based on randomly selected training samples, FP first outputs a measurement of the current model parameters (i.e., loss); with which, BP then updates these model parameters aiming at reducing the loss. In the context of synchronous distributed training, to guarantee convergence, workers synchronize their updated models periodically, e.g., by averaging their model updates/gradients, and a worker could not conduct the new FP until it obtains the updated global model; thus predictable and delay-sensitive transfers (i.e., *communication*) are triggered. Recent studies show that, by leveraging the layer-wise structure of deep models, the involved computation can partially overlap with the communication using pipelining [1]. Following this, AQGB tries to maximize these overlaps by transmitting quantized gradients and making work-conserving usage of available link capacities by adaptively adjusting the level of quantization/compression.

Overview. Currently, AQGB is specialized in peer-to-peer distributed training, in which a worker immediately broadcasts its quantized gradients to all other workers; then, by aggregating all the received gradients (including these generated by itself) and applying the results to its local model, each worker obtains the newly global updated model and move to the next round of training [2]. At the high level, by observing the patterns of how gradients are generated and consumed locally, AQGB could estimate the best completion time for the broadcast of its local gradients. Then, at the low level, during the broadcasting, based on both the un-delivered gradients and those will be available in the predictable near future, AQGB dynamically changes the level of quantization to react.

Next, we first sketch our flexible encode/decode scheme that enables AQGB to support multi-level quantization, then briefly introduced the principles AQGB uses to adjust quantization ratios.

Efficient multi-level quantization. In consideration that gradients in deep learning are generally float32 values (FP32 for short), as Figure 1 shows, AQGB designs an efficient multi-level quantization scheme based on the specific layout of FP32 used in computers. More specifically, according to the FP32 standard, a single-precision floating-point number takes up a total of 32 bits, which are divided into three parts: 1) 1 bit for the sign; 2) 8 bits for the exponent; 3) 23

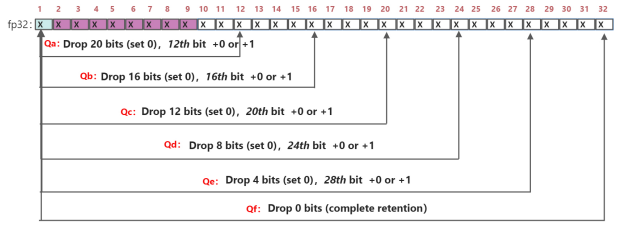
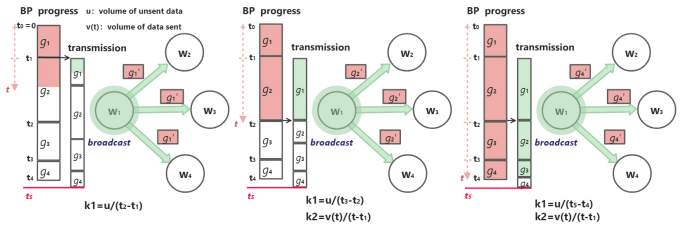


Figure 1: The multi-level quantization adopted by AQGB.



(a) At time t_1, g_1 is generated, $u=g_1$, $v(t_1)=0$.

(b) At time t_2, g_1 has mostly sent, $u=(g_1+g_2)$ - $v(t_2)$.

(c) At time t_4, g_1、g_2、g_3 has mostly sent, $u=(g_1+g_2+g_3+g_4)-v(t_4)$.

Figure 2: Examples showcasing how the adaptive quantization ratio control algorithm works.

bits for the mantissa. Distinguished from existing static quantization schemes, AQGB reduces the mantissa bits to achieve quantization on-demand, yielding efficient multi-level quantization.

As Figure 1 shows, the proposed quantization scheme supports 6 quantization levels, with the compression ratio of 0.375, 0.5, 0.625, 0.75, 0.875, and 1, respectively. During the broadcast, quantized gradients are encapsulated as UDP payloads. Given an array or a stream of FP32 gradient values, AQGB splits them into partitions and compresses gradients in the same partition with the same level of quantization. Thus, there are a total of 6 types of partitions. AQGB ensures that these partitions are of the same size after quantization. To support more fine-grained compression at the level of UDP, given a targeted compression ratio, AQGB tries to approximate it by selecting the numbers of different types of partitions properly.

Adaptive quantization ratio control. Now, we explain how AQGB controls the quantization ratio adaptively with examples. Consider that $=$ workers $F_1 \cdot \cdots \cdot F_=$ are training a neural network involving $<$ layers, a worker (e.g., F_1) starts its BP at $\mathbb{C}_0$ and would obtain the δ-th gradient blocks with the sizes of 6_δ at time $\mathbb{C}_\delta$. To maximize the overlapping between communication and computation, the worker wants the delivery of 6_δ to complete before $6_{\delta+1}$ becomes available, and the delivery of the final block $6_<$ to complete before the deadline of $\mathbb{C}_{<+1}$. Figure 2 shows a simple example, where $= = < = 4$. Let D and $E(\mathbb{C})$ be the original size of the worker's generated-yet-undelivered and delivered gradients at time $\mathbb{C}$, and $\delta(\mathbb{C})$ be the index of gradient that will appear next. Then, to achieve the above goal, the worker's broadcast rate should not be lower than $\because_1 = \frac{D}{\mathbb{C}_{\delta(\mathbb{C})} - \mathbb{C}_{\delta(\mathbb{C})-1}}$. However, from $\mathbb{C}_1$ to now, this worker's actual broadcast rate in average is $\because_2 = \frac{E(\mathbb{C})}{\mathbb{C}-\mathbb{C}_1}$. To eliminate this mismatch, the expected compression ratio that AQGB should approximate to quantization can be estimated as $\frac{\because_2}{\because_1}$.

(a) How the quantization ratio changes with bandwidth.

(b) The percent of communication overlapped with computation.

Figure 3: By adjusting the quantization ratio respecting the network state dynamically, AQGB greatly increases the overlapping between the communication and computation involved in distributed deep learning.

3 PRELIMINARY RESULTS

Settings. We develop an event-driven simulator with Python 3 to verify the performance of AQGB. We consider the example shown in Figure 2 in our tests, and assume that all workers are connected to the same switch with 1Gbps bidirectional links; broadcast/multicast transfers here would make work-conserving usage of the network fairly. Regarding $6_1 \cdot 6_2 \cdot 6_3 \cdot 6_4$ and $\mathbb{C}_1 \cdot \mathbb{C}_2 \cdot \mathbb{C}_3 \cdot \mathbb{C}_4 \cdot \mathbb{C}_5$, their values are 40MB, 20MB, 30MB, 10MB and 0.27s, 0.41s, 0.61s, 0.68s, 0.765s, respectively.

Results. As Figure 3a shows, with multi-level quantization and adaptive quantization ratio control, AQGB can dynamically change its compression ratio respecting the available bandwidth. Compared with the original pipelining design without adaptive gradient quantization, such a design further improves the percent of communication time that is overlapped with computation in a round of synchronization, from about 51″5% to about 98″1%. As it is impossible to achieve 100% overlapping in theory, such a result implies that AQGB could achieve near-optimal performances.

4 CONCLUSION

We have proposed AQGB, an Adaptive Quantized Gradient Broadcast scheme, to achieve approximate gradient synchronization for peer-to-peer distributed training. With an efficient multi-level quantization design, AQGB can dynamically adjust its level of quantization respecting the network state and the training's communication patterns, greatly increasing the overlap between the communication and computation for deep distributed learning [1].

REFERENCES

[1] Yanghua Peng, Yibo Zhu, Yangrui Chen, Yixin Bao, Bairen Yi, Chang Lan, Chuan Wu, and Chuanxiong Guo. 2019. A Generic Communication Scheduler for Distributed DNN Training Acceleration. In *27th SOSP* (Huntsville, Ontario, Canada). ACM, New York, NY, USA, 16–29.

[2] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. 2018. Error Compensated Quantized SGD and its Applications to Large-scale Distributed Optimization. In *35th ICML*, Vol. 80. PMLR, 5325–5333.

[3] Hang Xu, Chen-Yu Ho, Ahmed M. Abdelmoniem, Aritra Dutta, El Houcine Bergou, Konstantinos Karatsenidis, Marco Canini, and Panos Kalnis. 2021. GRACE: A Compressed Communication Framework for Distributed Machine Learning. In *41st ICDCS*. 561–572.