ON ESTIMATING OF EXECUTION TIMES FOR DISTRIBUTED LEARNING

A PREPRINT

Karimullin Timur
Russia, MIPT

Stanko Sergey
Russia, MIPT

ABSTRACT

Keywords Ok-TopK sparsifier · MARINA method · EF-21 method

The plan of the narrative will be approximately as follows: first we will introduce the main class of compressors, with which it will be convenient to work later. Then we will take apart some of specific compressors, give them definitions and formulate for some of them estimates of constants associated with them. After that we will consider the MARINA method in connection with any, generally speaking, sparsifier of a certain class and pay attention to its complexity in the simplest case. Next, we will consider the MARINA method with a special class of compressors, PermK, and give formulations for the convergence estimates of such a combination of techniques. Finally, we will observe the last method, EF-21, and formulate convergence estimates for this method applied together with the biased sparsifier TopK. Finally, we will compare the results in a single table.

1 Compressors and constants associated with them

We will recall some of the results presented in (Beznosikov et al. [2020]) and (Szlendak et al. [2021]).

Definition 1($B^3(\delta)$ compressors): We say that $Q \in B^3(\delta)$ for some $\delta > 0$ if:

$$E\left[\|Q(x) - x\|_2^2\right] \leq \left(1 - \frac{1}{\delta}\right)\|x\|_2^2, \quad \forall x \in R^d$$

Note that this is not the only class of compressors usually being observed in most papers, but the most convenient in analysis. Next, we introduce few specific compressors, or sparsifiers. First one is PermK, it has two versions of it, depending on internal values of n and d:

Definition 2.1(PermK for $d \geq n$): Assume that $d \geq n$ and $d = qn, q \in \mathbb{N}$. Let $\pi = (\pi_1, ..., \pi_d)$ be a random permutation of $\{1, ..., d\}$. Then for all $x \in \mathbb{R}^d$ we define:

$$Q_i(x) \coloneqq n \cdot \sum_{j=q(i-1)+1}^{qi} x_{\pi_j} e_{\pi_j}$$

Definition 2.2(PermK for $n \geq d$): Assume that $n \geq d, n > 1, n = qd, q \in \mathbb{N}$. Define multiset $S = \{1, ..., 1, 2, ..., 2, ..., d, ..., d\}$, where each number occurs precisely q times. Let $\pi = (\pi_1, ..., \pi_n)$ be a random permutation of S. Then for all $x \in \mathbb{R}^d$ and each $i \in \{1, 2, ..., n\}$ we define:

$$Q_i(x) \coloneqq d x_{\pi_i} e_{\pi_i}$$

Definition 3(RandK): For $k \in [d] \coloneqq \{1, \cdots, d\}$, for all $x \in R^d$ we define:

$$Q(x) \coloneqq \frac{d}{k} \sum_{i \in S} x_i e_i$$

Where $S \subseteq [d]$ is the k-nice sampling; i.e., a subset of $[d]$ of cardinality k chosen uniformly at a random, and $e_1, \cdots, e_d$ are the standard unit basis vectors in R^d.

Definition 3(TopK): For $k \in [d] \coloneqq \{1, \cdots, d\}$, for all $x \in R^d$ we define:

$$Q(x) \coloneqq \sum_{i=d-k+1}^{d} x_{(i)} e_{(i)}$$

Where coordinates are ordered by their magnitudes so that $|x_{(1)}| \leq |x_{(2)}| \leq \cdots \leq |x_{(d)}|$.

Now, although it is not well-known what the constants δ and ω are equal to for PermK sparsifier, we note the next two lemmas:

Lemma 1(δ and ω estimation for RandK): δ constant from Definition 1 for RandK sparsifier can be estimated as $\delta = \frac{1}{2-\frac{d}{k}}$. $\omega \coloneqq 1 - \frac{1}{\delta} = \frac{d}{k} - 1$ for RandK sparsifier.

Lemma 2(δ and ω estimation for TopK): δ constant from Definition 1 for TopK sparsifier can be estimated as $\delta = \frac{d}{k}$. $\omega \coloneqq 1 - \frac{1}{\delta} = 1 - \frac{k}{d}$ for TopK sparsifier.

These two lemmas are not very hard and proven in details in (Beznosikov et al. [2020]). We will need both for evaluating number of communications in MARINA models working in tandem with RandK and TopK sparsifier.

2 Introduction to the MARINA method and evaluation of its main characteristics

Now we refer to the article (Gorbunov et al. [2022]), where the MARINA method for distributed learning was first described. In a nutshell, the simplest variation of MARINA method works as follows: first, the server broadcasts the current g^k vector to all workers. Next, the iteration development option being chosen randomly: first scenario with probability p and second with probability $1-p$. At first scenario, all the workers count their next step point as $x^{k+1} = x^k - \gamma g^k$ and then set their local g_i^{k+1} value as $g_i^{k+1} = \nabla f_i(x^{k+1})$. At the second scenario, next step is being counted as $x^{k+1} = x^k - \gamma g^k$ and then g_i^{k+1} is being chosen as $g_i^{k+1} = g_i^k + Q\left(\nabla f_i(x^{k+1}) - \nabla f_i(x^k)\right)$. After either of those, g_i^{k+1} is being transmitted to a server, and server does an aggregation via $g^{k+1} = \frac{1}{n}\sum_{i=1}^n g_i^{k+1}$. The output - $\widehat{x}^K$ in the end is being chosen randomly from $\{x\}_{k=0}^{K-1}$.

Before we show complexity results for this algorithms, several assumptions need to be made. Originally, in (Gorbunov et al. [2022]) the only assumptions that have been made are that all n functions satisfy the gradient Lipschitz condition (and, thus, they all are differentiable of course), and that the total function f is bounded from below. To each function f_i was assigned its own gradient-lipschitz constant, L_i, and the value L was called $L^2 = \frac{1}{n}\sum_{i=1}^n L_i^2$.

Using an easily provable lemma showing a similar to a convexity property for a function, and using lemmas from classical probability theory in (Gorbunov et al. [2022]) authors proof the following theorem under the assumptions above:

Theorem 1(The simplest estimates for the MARINA method): After

$$K = O\left(\frac{\triangle_0 L}{\epsilon^2}\left(1 + \sqrt{\frac{(1-p)\omega}{pn}}\right)\right)$$

iterations with unbiased compressor with fixed ω value (from lemma 1), $\triangle_0 = f(x^0) - f_*$ and stepsize $\gamma \leq L^{-1}\left(1 + \sqrt{\frac{(1-p)\omega}{pn}}\right)^{-1}$, MARINA produces point $\widehat{x}^K$ for which $\mathbf{E}\left[\|\nabla f(\widehat{x}^K)\|^2\right] \leq \epsilon^2$. Moreover, if ζ_C is the expectation of the stored vector components when it is compressed, then under an assumption that the communication cost is proportional to the number of non-zero components of transmitted vectors from workers to the server, we have that the expected total communication cost per worker equals:

$$O\left(d + \frac{\triangle_0 L}{\epsilon^2}\left(1 + \sqrt{\frac{(1-p)\omega}{pn}}\right)(pd + (1-p)\zeta_C)\right)$$

Let us now introduce some more definitions regarded to Lipschitz constant of a vector-function, which would be useful further and were introduced in (Szlendak et al. [2021]):

Definition 4(L_-): Let $f_1, \cdots, f_n$ be differentiable. For $f(x) \coloneqq \frac{1}{n}\sum_{i=1}^n f_i(x)$ we understand L_- as the smallest non-negative number such that:

$$\|\nabla f(x) - \nabla f(y)\|^2 \leq L_-^2 \|x-y\|^2 \; for \; all \; x, y \in R^d$$

Definition 5(L_+): Let $f_1, \cdots, f_n$ be differentiable. We understand L_+ as the smallest non-negative number such that:

$$\frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq L_+^2 \|x-y\|^2 \; for \; all \; x, y \in R^d$$

Definition 6($L_\pm$): Finally, for differentiable $f_1, \cdots, f_n$ we understand $L_\pm$ as the smallest non-negative number such that:

$$\frac{1}{n}\sum_{i=1}^n ||\nabla f_i(x) - \nabla f_i(y)||^2 - ||\nabla f(x) - \nabla f(y)||^2 \leq L_\pm ||x-y||^2 \; for \; all \; x, y \in R^d$$

In the terms of new definitions, in (Szlendak et al. [2021]) authors point out that the result obtained in (Gorbunov et al. [2022]) can be actually improved trivially (a simple repetition of existing proof) to a:

$$K = O\left(\frac{\triangle_0}{\epsilon}\left(L_- + L_+\sqrt{\frac{(1-p)\omega}{pn}}\right)\right)$$

(No embarrassment should arise here: the authors in (Szlendak et al. [2021]) wish to obtain a solution with precision of ϵ instead of precision ϵ^2 in (Gorbunov et al. [2022]))

That is exactly the estimate we see in the table 2. More than that, the expected total communication cost per worker can be rewritten in the same manner, using L_- and L_+ terminology and with substitution of $\omega = \frac{d}{k} - 1$ in case of RandK compressor. Indeed, we have:

$$O\left(d + \frac{\triangle_0}{\epsilon}(pd + (1-p)k)\left(L_- + \sqrt{\frac{1-p}{p}\frac{\frac{d}{k}-1}{n}}L_+\right)\right)$$
$$= O\left(\frac{\triangle_0}{\epsilon}(p + (1-p)\frac{k}{d})\left(dL_- + \sqrt{\frac{1-p}{p}\frac{d}{k} - 1}\frac{dL_+}{\sqrt{n}}\right)\right)$$

Which, after a few transformations, can be written as:

$$\geq O\left(\frac{\triangle_0}{2\epsilon}\min\{dL_-, L_- + \frac{dL_+}{\sqrt{n}}\}\right) \geq O\left(\frac{\triangle_0}{2\epsilon}\min\{dL_-, \frac{dL_+}{\sqrt{n}}\}\right) (in \; case \; when \; d \geq n)$$

Which is exactly what one can see in the table 1.

3 Evaluation of MARINA + PermK combination method characteristics

Now, when dealt with MARINA + RandK method, let us turn to the article of (Szlendak et al. [2021]) closer, and describe how to obtain estimates for the method of MARINA + PermK.

Definition 7(AB inequality): We say that $\{Q_i\} \in \mathbb{U}(A, B)$ if there exist constants $A, B \geq 0$ such that the random operators $Q_1, ..., Q_n : \mathbb{R}^d \to \mathbb{R}^d$ satisfy the inequality:

$$\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^n Q_i(a_i) - \frac{1}{n}\sum_{i=1}^n a_i\right\|^2\right] \leq A\left(\frac{1}{n}\sum_{i=1}^n ||a_i||^2\right) - B\left\|\frac{1}{n}\sum_{i=1}^n a_i\right\|^2$$

In the article authors construct new compressors using random permutations (also referenced as PermK, see above).

To make it easier to classify compressor operators authors also use following definition:

Definition 8($\mathbb{IV}(A)$): We say that a collection $\{Q_i\}_{i=1}^n$ of unbiased operators form an input variance compressor system, if the variance of $\frac{1}{n}\sum_i Q_i(a_i)$ is controlled by a multiple of the variance of the input vectors $\{a_i\}_{i=1}^n$. That is, if there exists a constant $C \geq 0$ such that:

Table 1: The iteration complexity for minimizing $f(x)$ for MARINA and EF

Method + Compressors	T = Communication rounds
MARINA $\cap\{C_i\}_{i=1}^n \in \mathbb{U}(A,B)$	$O(\frac{\triangle_0}{\epsilon}(L_- + \sqrt{\frac{1-p}{p}((A-B)L_+^2 + BL_\pm^2)}))$
MARINA $\cap C_i \in \mathbb{U}(w)$ and independent	$O(\frac{\triangle_0}{\epsilon}(L_- + \sqrt{\frac{1-p}{p}\frac{w}{n}}L_+))$
EF21 $\cap C_i$ are α - contractive	$O(\frac{\triangle_0}{\epsilon}(L_- + (\frac{1+\sqrt{1-\alpha}}{\alpha} - 1)L_+))$

$$\mathbb{E}[||\frac{1}{n}\sum_{i=1}^n C_i(a_i) - \frac{1}{n}\sum_{i=1}^n a_i||^2] \leq C\text{Var}(a_1, ..., a_n)$$

for all $a_1, ..., a_n \in \mathbb{R}^d$. If this conditions are satisfied, we will write $\{Q_i\}_{i=1}^n \in \mathbb{IV}(C)$
In terms of A, B inequality this means, that if $\{Q_i\}_{i=1}^n \in \mathbb{U}(A,B)$ and $A = B$, then $\{Q_i\}_{i=1}^n \in \mathbb{IV}(A)$

Then, authors prove the correctness of these definitions (that they are suitable for assumptions that we mentioned in the beginning). For instance, PermK compressors from definitions 2.1 are unbiased and belong to $\mathbb{IV}(1)$, and PermK compressors from definition 2.2 are unbiased and belong to $\mathbb{IV}(1 - \frac{n-d}{n-1})$.

It is not hard to prove that, for example, for compressors from definition 8:

$$E[Q_i(x)] = n \cdot \sum_{j=q(i-1)+1}^{qi} \mathbb{E}[x_{\pi_j} e_{\pi_j}] = n(\sum_{j=q(i-1)+1}^{qi} \frac{1}{d}\sum_{i=1}^d x_i e_i) = \frac{nq}{d}x = x$$

Thus, compressors are unbiased.
Similarly, we can find the second moment:

$$\mathbb{E}[||Q_i(x)||^2] = n^2 \sum_{j=q(i-1)+1}^{qi} \mathbb{E}[|x_{\pi_j}|^2] = n^2 \sum_{j=q(i-1)+1}^{qi} \frac{1}{d}\sum_{i=1}^d |x_i|^2 = n^2\frac{q}{d}||x||^2 = n||x||^2 \tag{1}$$

And finally we can find A and B:

$$\mathbb{E}[||\frac{1}{n}\sum_{i=1}^n Q_i(a_i)||^2] = \frac{1}{n^2}\sum_{i=1}^n \mathbb{E}[||Q_i(a_i)||^2] + \sum_{i\neq j}\mathbb{E}[< Q_i(a_i), Q_j(a_j) >] \leq \frac{1}{n^2}\sum_{i=1}^n \mathbb{E}[||Q_i(a_I)||^2] = \frac{1}{n}\sum_{i=1}^n ||a_i||^2 \tag{2}$$

Thus $A = B = 1$ for compressors from definition 2.1.
Similarly, $A = B = 1 - \frac{n-d}{n-1}$ for compressors from definition 2.2.

So, to complete the refining of MARINA authors introduce the new quantity - Hessian variance, which is defined, as the the smallest quantity $L_\pm$ such that:

$$\frac{1}{n}\sum_{i=1}^n ||\nabla f_i(x) - \nabla f_i(y)||^2 - ||\nabla f(x) - \nabla f(y)||^2 \leq L_\pm ||x-y||^2$$

With $L_\pm, L_+$ and Lipschitz continuity L_- authors make an estimate for communication complexity for different methods and compressors. The results are represented in the table 1.

But after optimizing the method, by using concrete compressors results and by substituting certain A and B for different situations, iteration complexity can be improved. It is represented in the table 2.

4 Error feedback characteristics evaluation

Let us now turn to the last line in the considered tables 1 and 2 in order to investigate the convergence of such a method provided by (Richtárik et al. [2021]) and find the constants in the found dependence.

The algorithm itself is closely related to Markov chains. Markov chains can be said to compute on each particular node and on the server computer approximating vectors g^k which are close in convergence to the gradients of vectors f^k.

Table 2: The communication complexity for minimizing $f(x)$ for MARINA and EF with specified compressors

	Communication complexity	
Method + Compressor	$d \geq n$	$d \leq n$
MARINA $\cap$ PermK	$O(\frac{\triangle_0}{\epsilon}\min\{dL_-, \frac{d}{n}L_- + \frac{d}{\sqrt{n}}L_\pm\})$	$O(\frac{\triangle_0}{\epsilon}\min\{dL_-, L_- + \frac{d}{\sqrt{n}}L_\pm\})$
MARINA $\cap$ RandK	$O(\frac{\triangle_0}{\epsilon}\min\{dL_-, \frac{d}{\sqrt{n}}L_+\})$	$O(\frac{\triangle_0}{\epsilon}\min\{dL_-, L_- + \frac{d}{\sqrt{n}}L_+\})$
EF21 $\cap$ TopK	$O(\frac{\triangle_0}{\epsilon}dL_-)$	$O(\frac{\triangle_0}{\epsilon}dL_-)$

Briefly speaking, the algorithm can be described as follows: first, each worker receives from the server the point of the following calculations - x^{k+1} (which is similar to the first step of MARINA method). Then, each of them finds their local gradient ($\nabla f_i(x^{k+1})$) at this point, subtracts the locally stored vetcor (g_i^k) from it, and compresses the result (gaining $c_i^k = Q\left(f_i(x^{k+1}) - g_i^k\right)$). After that, each worker sends the result (vector c_i^k) back to the server and locally saves the updated local vector (gaining g_i^{k+1}) - the sum of the already stored vector and the compressed difference (i.e., $g_i^{k+1} = g_i^k + C\left(f_i(x^{k+1}) - g_i^k\right)$). At the end of the iteration, the server aggregates the result by updating its local vector (from g^k to g^{k+1}) with those vectors (compressed) that the workers sent to it: $g^{k+1} = g^k + \frac{1}{n}\sum_{i=1}^n c_i^k$.
Now let $\theta = 1 - \sqrt{1-\omega},\ \beta = \frac{1-\omega}{1-\sqrt{1-\omega}}$, where ω is taken from lemma 1. We will make use of these constants later.
In (Richtárik et al. [2021]) authors used only the assumption of f_i - functions having L_i - Lipschitz gradient, and for a described algorithm proved a following theorem in the most trivial case:
Theorem 2(EF21 convergence properties): After K iterations with, generally speaking, biased operator having fixed θ and β values, and having $\triangle_0 = f(x^0) - f_*$, if $\gamma \leq \left(L_- + L_+\sqrt{\frac{\beta}{\theta}}\right)$, then:

$$\mathbf{E}\left[\|\nabla f(x^K)\|^2\right] \leq \frac{2\triangle_0}{\gamma K} + \frac{\mathbf{E}\left[\frac{1}{n}\sum_{i=1}^n \|g_i^0 - \nabla f_i(x^0)\|^2\right]}{\theta K}$$

(We do have to make Q - compression in the very beginning, thus an expectation in the estimation)

As can be easily seen, in this case we do not have to do any assumptions on the correlation between values n and k, and have the same estimation for both of the cases. Using ω known for TopK sparsifier ($\omega = 1 - \frac{d}{k}$), we can obtain the result of $K = O(\frac{\triangle_0}{\epsilon}dL_-)$ (the main observation here is that by Jensen inequality $L_- \leq L_+$ holds). The very same result one can see in tables 1 and 2.

5 Presenting grades from the table 2 into a comparable form

In this section, we will give estimates for the time complexity of the algorithms from the table 2 in a form that can be further compared with the estimates obtained from the EF21-BC (Fatkhullin et al. [2021]) method.

First, we need to rewrite $\triangle_0$ from the table 2 in a different form for reasons that will become clear later. We will use the following simple inequality, which holds for convex and L - Lipschitz function f:

$$\forall x, y \in R^d \to 0 \leq f(y) - f(x) - < \nabla f(x), y - x > \leq \frac{L}{2} \parallel y - x \parallel^2$$

As was mentioned, firstly, L is the Lipschitz constant of the gradient of the function f, and hence in our established notation $L = L_-$. Second, the function f must be convex, which is not true in our case, but we can say that in the convergence zone (locally) the function is convex, which allows us to give the following estimate:

$$\triangle_0 = f_0 - f^* = f(z^0) - f(z^*) \leq < \nabla f(z^*), z^0 - z^* > + \frac{L_-}{2} \parallel z^0 - z^* \parallel^2 = 0 + \frac{L_-}{2} \parallel z^0 - z^* \parallel^2 = \frac{L_- R^2}{2}$$

Where $R^2 = \parallel z^0 - z^* \parallel^2$.

Our next step is to understand how to convert the expectation of minimum number of iterations for convergence of each algorithm from the table 2 into the expectation of minimum running time of each algorithm from the same table.

Although in the original MARINA method workers communicated with the server directly, we can no longer use the server, but let the machines exchange the full values of the compressed vectors under the PermK or RandK. Let us estimate the time complexity of such communications. For this purpose we multiply the asymptotic number of iterations of every method by the response time of the AllGather$v$ (Chan et al. [2007]) communication algorithm, where we

Table 3: Upper estimates on the expectation of minimum running time of the methods using MARINA and EF21 (case d $\leq$ n is of no interest)

Method $\cap Compressor$	Time complexity
MARINA $\cap$ PermK	$O(\frac{L_-^2 R^2}{\epsilon}\min\{1, \frac{1}{n} + \frac{L_\pm}{\sqrt{n}}\} \cdot d)$
MARINA $\cap$ RandK	$O(\frac{L_-^2 R^2}{\epsilon}\min\{1, \frac{L_\pm}{\sqrt{n}}\} \cdot d)$
EF21 $\cap$ TopK	$O(\frac{L_- L_+ R^2}{\epsilon} \cdot d)$

Table 4: Communication algorithms referenced in (Li and Hoefler [2022]) and time estimates for each one

Algorithm	Bandwidth	Latency
Dense	$2d\frac{n-1}{n}\beta$	$2 \cdot \log n\alpha$
TopA	$2k(n-1)\beta$	$\log n\alpha$
TopDSA	$\left[4k\frac{n-1}{n}\beta, (2k+d)\frac{n-1}{n}\beta\right]$	$(n + \log n)\alpha$
gTopK	$4k \cdot \log n \cdot \beta$	$2 \cdot \log n \cdot \alpha$
GaussianK	$2k(n-1)\beta$	$2 \cdot \log n \cdot \alpha$
Ok-TopK	$\left[2k\frac{n-1}{n}\beta, 6k\frac{n-1}{n}\beta\right]$	$(2n + 2\log n)\alpha$

substitute the number of transferred coordinates with its expectation, an example for MARINA $\cap$ PermK combination would look as follows:

$$O\left(\frac{\triangle_0}{\epsilon}\left(L_- + \sqrt{\frac{1-p}{p} \cdot 1 \cdot L_\pm}\right) \cdot 2 \cdot \frac{n-1}{n} \cdot \left(pd + (1-p)\frac{d}{n}\right)\beta\right)$$

As we can see, this repeats exactly the calculations for the results in table 2, only the factors $\frac{n-1}{n}$ and β appear: the first can be neglected in favor of constant, as well as the second. As a result, we obtain estimates from table 3.

However, the EF21 $\cap$ TopK combination is somewhat different. Note that error storage in one-way compression can be implemented locally, on each worker. To do this, in the EF21 algorithm, after compressing gradients, we will exchange compressed information in the DENSE manner, and then, when each worker has accumulated compressed vectors from every other worker, each worker will average the obtained vectors, and proceed to the next iteration of the algorithm.

Another interesting feature of the EF21 algorithm, not mentioned by the authors of Szlendak et al. [2021], is that $\alpha = 1$ when using the TopK compressor is only achievable when $K = d$, which will affect negatively the time of each communication. To demonstrate this, let us calculate the time complexity of the combination EF21 + TopK more accurately (the result can be seen in tables 3 and 5):

$$O\left(\frac{\triangle_0}{\epsilon}\left(L_- + \left(\frac{1+\sqrt{1-\alpha}}{\alpha} - 1\right)L_+\right) \cdot 2 \cdot \frac{n-1}{n} \cdot k \cdot \beta\right) = (since\ L_- \leq L_+) =$$
$$= O\left(\frac{L_- R^2}{\epsilon} \cdot \frac{d}{k} L_+ \cdot k \cdot \beta\right) = O\left(\frac{L_- L_+ R^2}{\epsilon} \cdot d \cdot \beta\right)$$

6 EF21 - BC

One of the most perspective mechanism for enforcing convergence of distributed gradient-based optimization methods enhanced with communication compression strategies based on the application of contractive compression operators is EF21 - BC (Fatkhullin et al. [2021]). It extends usage of EF21 to Stohastic approximation, enhancing EF21 with variance reduction, partial participation, momentum, but to evaluate Ok-TopK time all what we need is a support of bidirectional compression.

An algorithm, which supports it, is called EF21-BC. At each iteration clients compute and send to master node $c_i^t = Q^{dev}(\nabla f_i(x^{t+1}) - \widetilde{g}_i^{\,t}))$, where Q^{dev} is workers' (device's) compressor operator and g_i is a gradient at $i-th$ worker. Then we update $\widetilde{g}_i^{\,t+1} = \widetilde{g}_i^{\,t} + c_i^t$ and the main difference between classical EF21 is that master node also uses this mechanism, it computes and broadcast to workers vector: $b^{t+1} = Q^{serv}(\widetilde{g}^{t+1} - g^t)$ and updates $g^{t+1} = g^t + b^{t+1}$, where $\widetilde{g}^{t+1} = \frac{1}{n}\sum\limits_{i=1}^{n} \widetilde{g}_i^{\,t}$. Vector g^t is contained by master(server) and workers.Therefore, the clients are able to update

Table 5: Upper estimates on the expectation of minimum running time of the methods using MARINA, EF21 and EF21 - BC

Method ∩ Compressor	Time complexity
MARINA ∩ PermK	$O(\frac{L_-^2 R^2}{\epsilon}\min\{1, \frac{1}{n} + \frac{L_\pm}{\sqrt{n}}\} \cdot d)$
MARINA ∩ RandK	$O(\frac{L_-^2 R^2}{\epsilon}\min\{1, \frac{L_\pm}{\sqrt{n}}\} \cdot d)$
EF21 ∩ TopK	$O(\frac{L_- L_+ R^2}{\epsilon} \cdot d)$
EF21 - BC ∩ Ok-TopK	$O(\frac{L_+ L_- R^2}{\epsilon} \cdot n \cdot d)$
EF21 - BC ∩ Dense	$O(\frac{L_+ L_- R^2}{\epsilon} \cdot d)$
EF21 - BC ∩ gTopK	$O(\frac{L_+ L_- R^2}{\epsilon} \frac{d}{k} \cdot d \cdot \log n)$
EF21 - BC ∩ GaussianK	$O(\frac{L_+ L_- R^2}{\epsilon} \cdot d)$ (best case)
EF21 - BC ∩ TopK A	$O(\frac{L_+ L_- R^2}{\epsilon} \cdot n \cdot d)$
EF21 - BC ∩ TopK DSA	$O(\frac{L_+ L_- R^2}{\epsilon} \cdot d)$
EF21 ∩ TopK(PermK)	$O(\frac{L_-^2 R^2}{\epsilon} \cdot k)$

it via using $g^{t+1} = g^t + b^{t+1}$ and compute $x^{t+1} = x^t - \gamma g^t$ once they receive b^{t+1}

As stated in the source article (Fatkhullin et al. [2021]), the described algorithm takes $O(\frac{L_+ L_- R^2}{\epsilon}\delta^{serv}\delta^{dev})$ iterations to gain ϵ - solution.

7 Another compression algorithms for bidirectional compression in non - convex case

7.1 DoubleSqueeze

This algorithm, developed by Hanlin Tang and Liu [2020] is suitable for non-convex bidirectional compression. It has following steps:

1. Each worker computes the local stochastic gradient
2. Each worker computes the error-compensated stochastic gradient and update the local error according to the compressed error-compensated stochastic gradient.
3. All workers send the compressed error-compensated stochastic gradient to the parameter server, then the parameter server average all of it and and update the global error-compensated stochastic gradient together with the global error
4. The parameter server sends the updated compressed error-compensated stochastic gradient to all workers. Then each worker updates its local model.

The iteration complexity is:

$$O(\frac{\triangle_0}{\epsilon^{\frac{3}{2}}})$$

Which is worse then EF21.

7.2 DORE

In this article by Xiaorui Liu and Yan [2019] propose a new algorithm, which they call double residual compression (DORE). The algorithm consists of following phases:

1. Each worker node sends the compressed gradient residual to the master node and updates its state
2. The master node gathers the compressed gradient residual from all worker nodes and recovers the averaged gradient based on its state
3. The master node applies gradient descent algorithms

4. The master node broadcasts the compressed model residual with error compensation to all worker nodes and updates the model

5. Each worker node receives the compressed model residual and updates its model

It is motivated by the fact, that gradient changes smoothly for the smooth functions, so each worker can keep a state variable to keep it's previous gradient information. As a result, a residual between new gradient and a state should decrease, but the model will change only slightly, as algorithm converges. Therefore, authors propose to compress the model residual such that the compression variance can be minimized and also well bounded.
It has the following iterations complexity:

$$O(\frac{\triangle_0}{\epsilon})$$

Which is not any better then EF21.

8 OK-TopK algorithm

In this section, we want to describe a more general OK-TopK algorithm than the one that was introduced in (Li and Hoefler [2022]). Our generalized method consist of the following phases. Firstly, we select local TopK_2 values (notice that K_2 here is arbitrary large with only constraint standing for the obvious $K_2 \leq d$). Then we do balanced split and reduce, after which i-th worker has c_i number of values (we use here the same operation that was described in (Li and Hoefler [2022]) and thus numbers c_i are chosen by the original description of global network consensus). Next, we select global TopK_1 values (here it is obvious that at least $K_1 \leq \frac{d}{n}$ as consensus edge decision would be to split regions equally), which is followed by doing a data packaging, after which we balance data, so that each worker has almost equal part of the gradient. Finally, we also use Allgather$v$, which implies each worker having a full compressed gradient. An estimated bandwidth of the proposed algorithm then obviously lays in the interval: $[2k_2\frac{n-1}{n}\beta, (2K_2 + 4K_1)\frac{n-1}{n}\beta]$ and its latency is: $(2n + 2logn)\alpha$ as explained in details in (Li and Hoefler [2022]).

We argue that the description of the algorithm result in terms of operator expression was given incorrectly by the authors of the original article. In (Li and Hoefler [2022]), the authors mistakenly assumed that $Q^{serv} = Q^{dev} = TopK$, which is a rather crude estimate. Indeed, it is impossible to achieve a vector consisting of almost d components, assuming such. TopK, which was assumed by (Li and Hoefler [2022]) and applied at the end of the operation will select only K components, and the rest will be lost.

However, to estimate minimum running time of EF21-BC $\cap$ Ok-TopK algorithm, we have to explore Q^{serv} and Q^{dev} operators carefully in order to define the lowest score of δ^{serv} and δ^{dev} so that we could substitute these values in the EF21-BC iteration complexity and gain the lower bound of EF21-BC $\cap$ Ok-TopK running time. Thus, first thing we need to do is to write down correct form of the result of the algorithm in terms of operators. The main idea here would be to look at what each machine had at the start of the communication algorithm ($x_1, x_2, \cdots, x_n$ vectors), and write down an expression that describes the vector that would be stored at each machine locally by the end. It can be written as follows:

$$\sum_{i=1}^{n} TopK_1 \left[\frac{1}{n} \sum_{j=1}^{n} Q_i \left\{ TopK_2(x_j) \right\} \right] = \sum_{i=1}^{n} TopK_1 \left\{ Q_i \left[\frac{1}{n} \sum_{j=1}^{n} TopK_2(x_j) \right] \right\}$$

Where operator Q_i selects the i-th c_i values of the vector, such that $c_1 + c_2 + \cdots + c_n = d$. Our current task is to determine Q^{serv} and Q^{dev} such that the following is true:

$$Q^{serv} \left[\frac{1}{n} \sum_{i=1}^{n} Q^{dev}(x_i) \right] = \sum_{i=1}^{n} TopK_1 \left\{ Q_i \left[\frac{1}{n} \sum_{j=1}^{n} TopK_2(x_j) \right] \right\} \tag{3}$$

First, let $Q^{dev} = TopK_2$. If we assume so, we can find an operator Q^{serv} such that equality above holds $\forall x_1, x_2, \cdots, x_n \in R^d \times R^d \times \cdots \times R^d$. Indeed, all we need to do is to go through all the possible sets of vec-

tors $x_1, x_2, \cdots, x_n$ and define $\tilde{x}, \bar{x}$ as:

$$\tilde{x} = \frac{1}{n}\sum_{i=1}^{n} Q^{dev}(x_i) = \frac{1}{n}\sum_{i=1}^{n} TopK_2(x_i)$$

$$\bar{x} = \sum_{i=1}^{n} TopK_1\left\{Q_i\left[\frac{1}{n}\sum_{j=1}^{n} TopK_2(x_j)\right]\right\}$$

Now, we do know that $Q^{serv}(\tilde{x}) = \bar{x}$, so we do have a ready recipe for the construction of operator Q^{serv}, the only obstacle that may arise is this: it may turn out that at some point we have to match the same vector $\tilde{x}$ to two different vectors - $\bar{x_1}$ and $\bar{x_2}$. We state that this would not happen, as Q^{serv} satisfies $Q^{serv}(\tilde{x}) = \sum_{i=1}^{n} TopK_1\{Q_i\tilde{x}\}$, which means that the result of applying operator Q^{serv} is uniquely defined on the given vector $\tilde{x}$.

Notice that Q^{serv} does not necessarily operates on all vectors from R^d, which becomes more obvious if we let $n \cdot k \leq d$: operator Q^{serv} is defined as some entity for which (3) holds $\forall x_1, x_2, \cdots, x_n \in R^d \times R^d \times \cdots \times R^d$. Next, we will show that:

$$\forall x_1, x_2, \cdots, x_n \rightarrow \|Q^{serv}(x) - x\|^2 = [where\ x = \frac{1}{n}\sum_{i=1}^{n} Q^{dev}(x_i) = \frac{1}{n}\sum_{i=1}^{n} TopK_2(x_i)] = \tag{4}$$

$$= \left\|Q^{serv}\left(\frac{1}{n}\sum_{i=1}^{n} TopK_2(x_i)\right) - \frac{1}{n}\sum_{i=1}^{n} TopK_2(x_i)\right\|^2 < \left\|\frac{1}{n}\sum_{i=1}^{n} TopK_2(x_i)\right\|^2$$

Which means that Q^{serv}, whatever that operator might be, is actually a compressor and δ^{serv} from Definition 1 can be found. After that we will show that x^0, $\tilde{\delta}^{serv}$ could be found such that:

$$b) \quad \left\|Q^{serv}(x^0) - x^0\right\|^2 \leq \left(1 - \frac{1}{\tilde{\delta}^{serv}}\right)\|x^0\|^2$$

8.1 a)

Let us make the following redefining:

$$y = Q^{serv}\left(\frac{1}{n}\sum_{i=1}^{n} TopK_2(x_i)\right) = \sum_{i=1}^{n} TopK_1\left\{Q_i\left[\frac{1}{n}\sum_{j=1}^{n} TopK_2(x_j)\right]\right\} x = \frac{1}{n}\sum_{i=1}^{n} TopK_2(x_i) \tag{5}$$

$$x = \frac{1}{n}\sum_{i=1}^{n} TopK_2(x_i) \tag{6}$$

Now, it is obvious that in terms of y, x denotations a) clause would be rewritten as:

$$\|y - x\|^2 = \|y\|^2 + \|x\|^2 - 2 < y, x > < \|x\|^2 \Rightarrow \|y\|^2 < 2 < y, x >$$

We can show that using a straight-forward approach: let $\bar{i_1} = i_{11}, i_{12}, \cdots, i_{1K_1}$ be the indexes of the result of taking Top$K_1(x_1)$; $\bar{i_2} = i_{21}, i_{22}, \cdots, i_{2K_1}$ be the indexes of the result of taking Top$K_1(x_2)$, and so on. Let $\bar{j}$ be set of indexes of vector x, that is, $\bar{j} = \bar{i_1} + \bar{i_2} + \cdots + \bar{i_n}$. Now, it is clear that the indices of the vector y fall either into $Q_1(\bar{j})$, either into $Q_2(\bar{j})$, etc. This means that vector y differs from vector x only in the fact that vector y zeros out some components of vector x, leaving the rest intact. Let's say the indices remaining in the vector y form the set $\bar{l}$, then it is clear that $|\bar{l}| \leq |\bar{j}|$, also, elements of y at positions from the set l coincide with elements of x at the same positions by virtue of the definition of y and x. This leads us to:

$$2 < y, x > -\|y\|^2 = 2\|y\|^2 - \|y\|^2 = \|y\|^2 > 0$$

We put a strong inequality sign in the expression above, since zero vectors of x are not fed into definition 1, which means that the components of y cannot all be zero at the same time. This ends the proof of point a.

8.2 b)

We obviously have to consider two different cases here: $nK_1 \leq d$ and $d \leq nK_1$, as examples of x^0 would differ in such cases. Considering first case, let x^0 be a vector of size d consisting of nK_1 ones. Then the obvious partition (which we can choose frivolously, as discussed above) of this vector into n pieces will be n vectors consisting of K_1 ones. It is then obvious that (in terms of (5), (6)) y will be a vector of K_1 ones, while x will be a vector consisting of nK_1 ones. And thus:

$$\|y - x\|^2 = ((n-1)K_1)^2 \leq \left(1 - \frac{1}{\delta}\right)\|x\|^2 = \left(1 - \frac{1}{\delta}\right) \cdot (nK_1)^2 \Rightarrow \delta \geq \frac{n^2}{2n-1} \geq \frac{n}{2}$$

Thus, the smallest $\tilde{\delta}^{serv}$ required by vector x^0 to make the corresponding estimate for it is $\frac{n}{2}$, where n is limited by the inequality $nK_1 \leq d$, which means that this minimum estimate at its maximum can be no more than $\frac{d}{K_1}$. The estimation of delta in the case $d \leq nK_1$ can be done in the same way, only with choosing not K_1, but $\frac{d}{n}$ ones.

Knowing maximum values of the minimum scores of δ^{dev} (as for workers, we simply have $Q^{dev} = TopK_2$ and thus $\delta^{dev} = \frac{d}{K_2}$) and δ^{serv}, we can estimate the minimum working time of EF21-BC ∩ Ok-TopK as follows:

$$O\left(\frac{L_+L_-R^2}{\epsilon} \cdot \frac{d}{K_2} \cdot \frac{d}{2K_1} \cdot [2K_2\frac{n-1}{n}\beta, (2K_2 + 4K_1)\frac{n-1}{n}\beta]\right)$$

Which is asymptotically equal to:

$$O\left(\frac{L_+L_-R^2}{\epsilon} \cdot \frac{d}{K_2} \cdot \frac{d}{2K_1} \cdot [2K_2\frac{n-1}{n}\beta, (2K_2 + 4K_1)\frac{n-1}{n}\beta]\right) = O\left(\frac{L_+L_-R^2}{\epsilon} \cdot \frac{d}{K_2} \cdot \frac{d}{K_1} \cdot K_2\right)$$
$$= O\left(\frac{L_+L_-R^2}{\epsilon} \cdot d \cdot \frac{d}{K_1}\right)$$

And finally:

$$= O\left(\frac{L_+L_-R^2}{\epsilon} \cdot d \cdot \frac{d}{K_1}\right) \geq [K_1 \leq \frac{d}{n}] \geq O\left(\frac{L_+L_-R^2}{\epsilon} \cdot d \cdot n\right)$$

Here we assumed that $K_2 > K_1$, however, the same evaluation can be achieved in the opposite case. This result indicates not only that the time asymptotic of EF21-BC ∩ Ok-TopK algorithm is worse than EF21-BC ∩ Dense algorithm by the factor of n, but also that the time asymptotic of EF21-BC ∩ Ok-TopK algorithm is worse than MARINA ∩ PermK and MARINA ∩ RandK algorithm most likely (if n is large enough) by the factor of $n^{\frac{3}{2}}$.

8.3 EF21-BC compatibility

It is worth noting that we have not discussed here the compatibility of the Ok-TopK communication exchange algorithm with the parallel EF21-BC gradient descent counting algorithm, but only obtained some estimates of characteristic constants of Q^{serv}, Q^{dev} for Ok-TopK in the case where the Ok-TopK algorithm can be implemented. But in fact, suppose we store on each node only a part of the vector $\tilde{g}^t$ storing the error. The size of this vector corresponds to the size of the region the machine is responsible for. Then it is clear that each node, having a stored part of vector $\tilde{g}^t$, can update its stored value $\tilde{g}^t$ by adding to the stored vector what was collected from other nodes in the "Split and Reduce" phase before executing part of "Balance and AllgatherV". But this means that the expression $Q^{serv}(\tilde{g}^{t+1} - g^t)$ can always be calculated, and with it the vector g^t can be updated and the loop can be started again, which means that the Ok-TopK algorithm is compatible with EF21-BC.

9 Other compressing operators

Let us now introduce other communication algorithms which were referenced in (Li and Hoefler [2022]). They are listed in table 4, where one can find time complexity for each of them. Note that all of them could be applied with the same EF21-BC too, with the help of the analysis that we made for Ok-TopK method.

9.1 Dense

Dense communication protocol can be briefly represented as Reduce + Broadcast protocol taken from (Chan et al. [2007]). Obviously, at the end of the Dense communication algorithm, each worker has a local accumulation of:

$$\frac{1}{n}\sum_{i=1}^{n} x_i$$

Where x_i - vector that was stored on worker number i at the beginning of the Dense algorithm. That means that in the notation of the EF21-BC algorithm we have $Q^{serv} = Id$, $Q^{dev} = Id$, which implies $\delta^{serv} = \delta^{dev} = 1$, and thus we obtain:

$$O\left(\frac{L_+L_-R^2}{\epsilon}\cdot 2d\frac{n-1}{n}\beta\right) = O\left(\frac{L_+L_-R^2}{\epsilon}\cdot d\right)$$

Which one can see in table 5

9.1.1 EF21-BC compatibility

Note that, asymptotically, the result of $EF21-BC+Dense$ running time does not depend in any way on whether we compress gradients and perform sparse data transfer or not.

Obviously, this communication algorithm is fully compatible with EF21-BC, because each node eventually accumulates the full sum $\sum_{i=1}^{n} c_i$ (see Section 6), which means that after this exchange nothing prevents each node from calculating an error received at this iteration and finding $\tilde{g}^{t+1} - g^t$, and thus, just putting $g^{t+1} = \tilde{g}^{t+1}$.

9.2 gTopK

Let us briefly explain the gTopK communication algorithm taken from the (Shi et al. [2019]). In the first iteration, each worker exchanges with its neighbor the result of the TopK sparsifier applied to its dataset. The neighbor accepts his reponse, and applies TopK to the sum of TopK applied to his own dataset and the response. This continues in pairs until the global value of TopK is obtained. We will do the analysis for the example of 4 workers, but it can easily be generalized to the case of any number of devices. Formally, if we set x_1, x_2, x_3, x_4 as vectors being contained at the worker 1, 2, 3 and 4, respectively, the result of such sparsification can be written as:

$$TopK\left\{TopK\left[TopK(x_1)+TopK(x_2)\right]+TopK\left[TopK(x_3)+TopK(x_4)\right]\right\}$$

We repeat the reasoning done for Ok-TopK and try to find Q^{serv}, Q^{dev}, such that:

$$Q^{serv}\left[\frac{1}{4}\sum_{i=1}^{4} Q^{dev}(x_i)\right] = TopK\left\{TopK\left[TopK(x_1)+TopK(x_2)\right]+TopK\left[TopK(x_3)+TopK(x_4)\right]\right\}$$

And again, let $Q^{dev} = TopK$. In such a case, as you one see, the uniqueness of the operator, and therefore the very fact of its existence, is no longer present. Indeed, suppose the vectors x_1, x_2, x_3, x_4 are such that $x_1^T = (6,6,6,0,0,0)$; $x_2^T = (0,0,0,5,5,5)$; $x_3^T = (4,4,4,0,0,0)$; $x_4^T = (0,0,0,3,3,3)$, then $gTopK = Q^{serv}\left[\sum_{i=1}^{4} TopK(x_i)\right] = (10,10,10,0,0,0)^T$. However, $\sum_{i=1}^{4} TopK(x_i) = (10,10,10,8,8,8)^T$. The set $x_1 = (6,6,6,0,0,0)$; $x_2 = (0,0,0,3,3,3)$; $x_3 = (4,4,4,0,0,0)$; $x_4 = (0,0,0,5,5,5)$, will return $Q^{serv}\left[\sum_{i=1}^{4} TopK(x_i)\right] = (6,6,6,0,0,0)$, so we are faced with a situation where the Q^{serv} operator needs to map the same vectors to the different values, and thus, formally, no such operator exists. However, getting rid of formalities, and extending the operator Q^{serv} to one that takes into account the filling of vectors x_1, x_2, x_3, x_4 and making the matched sums in the example $\sum_{i=1}^{4} Q^{dev}(x_i)$ sort of different, the problem of non-existence of such operator can be omitted.

Again, let us define $\tilde{x}$, $\bar{x}$ as:

$$\tilde{x} = \frac{1}{4}\sum_{i=1}^{n} Q^{dev}(x_i) = \frac{1}{n}\sum_{i=1}^{n} TopK(x_i) = \frac{1}{4}\sum_{i=1}^{4} TopK(x_i)$$

$$\bar{x} = Q^{serv}\left[\frac{1}{4}\sum_{i=1}^{4} Q^{dev}(x_i)\right] = TopK\left\{TopK\left[TopK(x_1)+TopK(x_2)\right]+TopK\left[TopK(x_3)+TopK(x_4)\right]\right\}$$

9.2.1 a)

We can show a similar property to point a in section 8 of the compression operator Q^{serv}:

$$\left\|Q^{serv}\left(\frac{1}{4}\sum_{i=1}^{4}TopK(x_i)\right)-\frac{1}{4}\sum_{i=1}^{4}TopK(x_i)\right\|^2<\left\|\frac{1}{4}\sum_{i=1}^{4}TopK(x_i)\right\|^2$$

Indeed, if we study how gTopK chooses components from vectors $x_1,\cdots,x_n$, then it becomes clear that either gTopK does not choose indices from $\tilde{x}$ at all, or it chooses them, but components of not all vectors are included in the result of gTopK (components of some vectors could be simply discarded in the process of gTopK selection), or it chooses indices from $\tilde{x}$, choosing all terms from $x_1,\cdots,x_n$ entirely at the taken index.

9.2.2 b)

A similar example to section 8 - x_0 can be given to estimate δ^{serv} from below, just considering either n vectors of k units (in case $nk\leq d$) or n vectors of $\frac{d}{n}$ units (more cunning example can be chosen, getting a larger value of δ, but this is enough for our purposes).

This makes us substitute $\delta^{serv}=\delta^{dev}=\frac{d}{k}$ and $4k\cdot\log n\cdot\beta$ - working time for gTopK algorithm into EF21-BC time complexity evaluation, giving gTopK the max discount we can (δ^{serv} - number is, generally speaking, larger than the one ($\frac{d}{k}$) we picked up, so that we thus occupy the undervalued complexity of the algorithm):

$$O\left(\frac{L_+L_-R^2}{\epsilon}\left(\frac{d}{k}\right)^2\cdot 4k\cdot\log n\cdot\beta\right)=O\left(\frac{L_+L_-R^2}{\epsilon}\frac{d}{k}\cdot d\cdot\log n\right)$$

Which is much worse than evaluation for MARINA ∩ PermK algorithm.

9.2.3 EF21 compatability

It is worth noting that this communication exchange algorithm is not applicable at all with the EF21-BC gradient descent counting algorithm, because no node throughout the exchange accumulates the sum $\sum_{i=1}^{n}c_i$ (see Section 6). However, its evaluation is important for calculating the error (vector $\tilde{g}^t$ from section 6). However, the obtained estimates do not lose their meaning, since we can interpret them in the following way: suppose that on each machine the vector $\tilde{g}^t$ is known, in this case the algorithm will be compatible. Further, we can say that each node knows about the value of such an aggregate sum "for free", which means that the estimates obtained here make sense as the lower estimates for the algorithms being used together with gTopK and error feedback mechanism (because EF21-BC is the fastest of the currently known gradient descent algorithms with compression in both directions).

9.3 Top$k$A

The main idea of this algorithm by (Renggli et al. [2019]) is using quantization operator Q on gradients, which were accumulated from other workers, who before compressed it with Top$k$. This algorithm represents the allgather based approach. It uses recursive doubling scheme in t-th iteration nodes from distance 2^{t-1} exchange all previously reduced $2^{t-1}k$ data items. The following algorithm gives us bandwith, equals to $k(P-1)$.
Therefore, the total operator, used in Top$k$A looks like:

$$C=Q(\tfrac{1}{n}\sum_{i=1}^{n}TopK(x_i))$$

The quantization operator compression can be found in (Alistarh et al. [2017]) and it's compression asymptotically equals to $\delta^{serv}=1$ and $TopK=Q^{dev}$ compression rate can be evaluated as $\delta^{dev}=\frac{d}{k}$.
Therefore the total time complexity is:

$$O(\frac{L_+L_-R^2}{\epsilon}\cdot 1\cdot\frac{d}{k}\cdot k\cdot(n-1))=O(\frac{L_+L_-R^2}{\epsilon}\cdot n\cdot d)$$

9.4 Top$k$DSA

This algorithm is very similar to Top$k$A, but is designed to be more suitable for large data case. So it represents the Dynamic Sparse Allreduce used in SparCML (Renggli et al. [2019]), which consists of reduce-scatter and allgather. So as in Top$k$A we use the quantization operator on server, and TopK operator on worker.
Therefore the total time complexity is:

$$O(\frac{L_+L_-R^2}{\epsilon}1 \cdot \frac{d}{k} \cdot k \cdot \frac{n-1}{n}) = O(\frac{L_+L_-R^2}{\epsilon}d)$$

9.5 GaussianK

The GaussianK communication exchange algorithm is not really applicable to distributed computing, or rather the formal approach to its description in the article referred to by the authors of (Li and Hoefler [2022]) does not apply, in case with GaussianK single-node mode being used originally. The estimate, which is obtained by the authors of (Li and Hoefler [2022]) in table 4 for the running time of such a communication algorithm actually comes from the fact that the authors of (Li and Hoefler [2022]) put the mechanism of such an exchange similar to Top$k$A.
The authors of the (Shaohui Shi and See [2019]) do not estimate the value of δ for GaussianK. Let us assume for now that it is just some value of δ. Total time complexity then is:

$$O(\frac{L_+L_-R^2}{\epsilon}\delta \cdot 1 \cdot k \cdot (n-1) = O(\frac{L_+L_-R^2}{\epsilon}\delta \cdot k \cdot n \cdot \frac{\frac{d}{k}}{\frac{d}{k}}) = O(\frac{L_+L_-R^2}{\epsilon}\frac{\delta}{\frac{d}{k}} \cdot n \cdot d)$$

For this algorithm to be at least better than EF21-BC + Dense, we would need $\frac{\delta}{\frac{d}{n \cdot k}} \leq 1$, which would entail $\delta = \frac{d}{n \cdot k}$, which is already a large reduction in the compression rate compared to ordinary TopK, which, according to the authors of the original paper (Shi et al. [2019]), is an unrealistic situation, since, as they claim, the δ value for GaussianK is little different from that for TopK.

10 Different approach to the definition of widely known compressors

10.1 TopK analysis

We will start with a study of the TopK compressor. As we know, this compressor does not multiply the components taken from the vector by the increasing constant. Let us try to determine it in another way, namely, we take away the top K components of the vector, and then we multiply the result by some constant a. In this case, as it is well known from (Beznosikov et al. [2020]), we obtain the following inequalities on such an operator (we will call it aTopK hereafter):

$$a^2 \cdot \frac{k}{d}\|x\|^2 \leq \|a \cdot TopK(x)\|^2 =< a \cdot TopK(x), a \cdot x > \leq a^2 \cdot \|x\|^2$$

Thus, we gain the following:

$$\begin{aligned} E\left[\|aTopK(x) - x\|^2\right] = \|aTopK(x) - x\|^2 = \|aTopK(x)\|^2 + \|x\|^2 - 2 < aTopK(x), x >= \\ =< aTopK(x), aTopK(x) > + \|x\|^2 - \frac{2}{a} < aTopK(x), a \cdot x >= \\ =< aTopK(x), a \cdot x > + \|x\|^2 - \frac{2}{a} < aTopK(x), a \cdot x >= \\ = \|x\|^2 - \left(\frac{2}{a} - 1\right) < aTopK(x), a \cdot x > \leq \|x\|^2 - \frac{2-a}{a} \cdot a^2 \cdot \frac{k}{d}\|x\|^2 = \\ = \|x\|^2 \left(1 - \frac{(2-a)a^2k}{ad}\right) \Rightarrow \delta = \frac{d}{(2-a)ak} \end{aligned}$$

From this we see that when a is not fixed, the smallest value of δ that we can extract from the aTopK operator is the familiar $\frac{d}{k}$, attainable at $a = 1$. Thus, changing the parameter a makes little sense and gives no improvement in performance.

10.2 PermK analysis

We investigate here only the case $d \geq n$, since it is of most interest to us. In the original paper PermK, as discussed above, works as follows: it chooses a permutation of the indices for which the worker who used the operator is responsible, and multiplies the chosen components by some constant a, which in the original paper is assumed to be n. Let us start with the fact that the authors of the (**?**) chose this value for a reason: the value $a = n$ allows the PermK operator to be an unbiased operator, which, in turn, allows it to be applied together with the MARINA algorithm. However, when EF21 is used, the biasedness of the operator does not play any great role, and we can change this parameter to suit our needs - in order to reduce the δ parameter of the operator.

As was shown in the aforementioned (**?**), for the operator aPermK (aPermK is a denotion of a PermK with an arbitrary constant a instead of n) the following equalities are satisfied:

$$E[PermK_i(x)] = a \cdot \sum_{j=q(i-1)+1}^{qi} E\left[x_{\pi_j} e_{\pi_j}\right] = a\left(\sum_{j=q(i-1)+1}^{qi} \frac{1}{d}\sum_{i=1}^{d} x_i e_i\right) = \frac{aq}{d}x = \frac{a}{n}x$$

$$E\left[\|PermK_i(x)\|^2\right] = a^2 \cdot \sum_{j=q(i-1)+1}^{qi} E\left[|x_{\pi_j}|^2\right] = a^2 \cdot \sum_{j=q(i-1)+1}^{qi} \frac{1}{d}\sum_{i=1}^{d} |x_i|^2 = \frac{a^2 q}{d}\|x\|^2 = \frac{a^2}{n}\|x\|^2$$

Now, let us perform an evaluation similar to the previous subsection for the operator aPermK:

$$E\left[\|aPermK_i(x) - x\|^2\right] = E\left[\|aPermK_i(x)\|^2\right] + E\left[\|x\|^2\right] - 2E\left[< aPermK_i(x), x >\right] =$$
$$= \frac{a^2}{n}\|x\|^2 + \|x\|^2 - 2\frac{a}{n}\|x\|^2 = \|x\|^2\left(1 - \left(\frac{2a - a^2}{n}\right)\right) \Rightarrow \delta = \frac{n}{2a - a^2}$$

As one can see, we do not get any improvement by varying the parameter a, because the minimum δ we get with this approach is $\delta = n$ for $a = 1$.

10.3 TopK(aPermK) operator

Now we will try to cross the two considered operators by combining them together. Namely, we will apply the $TopK$ operator to the result returned by the $aPermK$ operator. Taking into account all equalities obtained in the previous subsections, we obtain (Here we replace $aPermK_i$ with $aPermK$ for ease of readability, implying that we make an estimate on the $TopK(aPermK_i)$ operator applied by the i-th worker):

$$E\left[\|TopK(aPermK(x)) - x\|^2\right] = E\left[\|TopK(aPermK(x))\|^2\right] + E\left[\|x\|^2\right] - 2E\left[< TopK(aPermK(x)), x >\right] =$$
$$= E\left[< TopK(aPermK(x)), TopK(aPermK(x)) >\right] + \|x\|^2 - 2E\left[< TopK(aPermK(x)), x >\right] =$$
$$= E\left[< TopK(aPermK(x)), aPermK(x) >\right] + \|x\|^2 - \frac{2}{a}E\left[< TopK(aPermK(x)), aPermK(x) >\right] =$$
$$= \|x\|^2 - \frac{2-a}{a}E\left[< TopK(aPermK(x)), aPermK(x) >\right] = \|x\|^2 - \frac{kn}{d}\frac{2-a}{a}E\left[\|aPermK(x)\|^2\right] =$$
$$= \|x\|^2 - \frac{kn}{d}\frac{2-a}{a}\frac{a^2}{n}\|x\|^2 = \|x\|^2\left(1 - \frac{k(2-a)a}{d}\right) \Rightarrow \delta = \frac{d}{k(2-a)a}$$

As one can see, $\delta = \frac{d}{k}$ is minimal and achievable when parameter $a = 1$. Thus, $TopK(aPermK(x))$ operator is no better when being used isolately in EF21.

10.4 aPermK(TopK) operator

Let us now consider mixing the $TopK$ and $aPermK$ operators in a different order: let us first apply $TopK$ on each worker locally, after which we will pass the selected K indices through the $aPermK$ mechanism. Using the equalities obtained in the subsections above, we obtain:

$$E\left[\|aPermK(TopK(x)) - x\|^2\right] = E\left[\|aPermK(TopK(x))\|^2\right] + E\left[\|x\|^2\right] - 2E\left[< aPermK(TopK(x)), x >\right] =$$
$$= E\left[\frac{a^2}{n}\|TopK(x)\|^2\right] + \|x\|^2 - 2E\left[\frac{a}{n} < TopK(x), x >\right] = \frac{a^2}{n}\|TopK(x)\|^2 + \|x\|^2 - \frac{2a}{n} < TopK(x), x >=$$
$$= \frac{a^2}{n} < TopK(x), x > + \|x\|^2 - \frac{2a}{n} < TopK(x), x >= \|x\|^2 - \left(\frac{2a - a^2}{n}\right) < TopK(x), x >\leq$$
$$\leq \|x\|^2 - \frac{2a - a^2}{n} \cdot \frac{k}{d}\|x\|^2 = \|x\|^2\left(1 - \frac{(2a - a^2)k}{nd}\right) \Rightarrow \delta = \frac{nd}{(2a - a^2)k}$$

Unfortunately, as one can see, with this approach the minimal δ we get is $\frac{nd}{k}$, which is n times larger than the usual δ for TopK. This, obviously, only worsens the estimation of the running time of the EF21 + TopK algorithm.

11 Convergence rate of TopK in a simple case

Here we consider TopK convergence in the case when $L_{\pm} = 0$ for the purpose of getting closer to TopK(PermK) operator.

In particular, we say $f_i = f = \frac{1}{2}x^T Ix$

$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\gamma_k\langle T(\nabla f(x^k), x^k - x^*\rangle + \gamma_k^2\|T(\nabla f(x^k)\|^2 = |x^* = 0, \nabla f(x^*) = 0, \nabla f(x^k) = Ix^k| = \|x^k\|^2 - 2\gamma_k\langle T(x^k), x^k\rangle + \gamma_k^2\|T(Ix^k)\|^2 = \|x^k\|^2 - 2\gamma_k\|T(x^k)\|^2 + \gamma_k^2\|T(x^k)\|^2 = \|x^k\|^2(\gamma_k^2 - 2\gamma_k)|\gamma_k = 1| \leq (1 - \frac{m}{n})\|x^k\|^2$

For the identity matrix: $\mu_I = 1, L_I = 1$, so we understand, that in more general case the multiplier will be $(1 - \frac{\mu m}{Ln})$, without any other constants (otherwise we would have got them here)

Which gives us the convergence rate:

$$\mathcal{O}\left(\frac{\Delta_0 dL}{\varepsilon m\mu}\right)$$

12 Comparison with EF21

According to Richtarik et al 2021, communication rounds for convergence using $\mathcal{C}_\alpha$ is

$$\mathcal{O}\left(\frac{\Delta_0}{\varepsilon}\left(L_- + \left(\frac{1 + \sqrt{1 - \alpha}}{\alpha} - 1\right)L_+\right)\right) \tag{7}$$

If the case of $L_{\pm} = 0$, $L_- = L_+$ holds. Thus leading to following rate.

$$\mathcal{O}\left(\frac{\Delta_0}{\varepsilon}\frac{1 + \sqrt{1 - \frac{m}{d}}}{\frac{m}{d}}L_+\right) = \mathcal{O}\left(\frac{\Delta_0}{\varepsilon}\frac{d + d\sqrt{1 - \frac{m}{d}}}{m}L_+\right) \tag{8}$$

13 A-B inequality for TopK(aPermK) sparsificator

Let us try to show A-B inequality under zero-variance conditions (meaning $a = a_1 = a_2 = \cdot = a_n = \sum_{i=1}^n a_i = \tilde{a} = \nabla f_1(x) = \nabla f_2(x) = \cdot = \nabla f_n(x) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x)$):

$$E\left[\| \left(C_i(a_i) - a_i\right) \|^2\right] = E\left[\| C_i(a_i) \|^2\right] +$$

$$+E\left[\| \sum_{i=1}^{n} a_i \|^2\right] - 2E\left[< \sum_{i=1}^{n} C_i(a_i); \sum_{i=1}^{n} a_i >\right] =$$

$$= E\left[\| \sum_{i=1}^{n} TopK(aPerm_iK(a_i))\|^2\right] + E\left[\| \sum_{i=1}^{n} a_i\|^2\right] -$$

$$2E\left[< \sum_{i=1}^{n} TopK(aPerm_iK(a_i))\|^2\right] + E\left[\| \sum_{i=1}^{n} a_i\|^2\right] -$$

$$2\sum_{i=1}^{n} E\left[< TopK(aPerm_iK(a_i)); \sum_{i=1}^{n} a_i >\right] = \cdots =$$

$$= \sum_{i=1}^{n} E\left[\|TopK(aPerm_iK(a))\|^2\right] + n^2E[\|\tilde{a}\|^2 -$$

$$\frac{2n}{a}\sum_{i=1}^{n} E\left[< TopK(aPerm_iK(a)); aPerm_iK(\tilde{a}) >\right] = \cdots =$$

$$= n \cdot \sum_{i=1}^{n} \|a\|^2 \left(\frac{a^2}{n^2} - \frac{ka}{nd}\right) - \left(\frac{kan}{d} - n^2\right) \|\tilde{a}\|^2 \Rightarrow$$

$$\Rightarrow A = \frac{a^2}{n^2} - \frac{ka}{nd} < 1;$$

$$B = \frac{kan}{d} - n^2 < 1 \Rightarrow only\ a\ that\ feasible\ is\ a > \frac{nd}{k},\ but\ :$$

$$A(a = \frac{nd}{k}) = \frac{d^2}{k^2} - 1 > 1$$

$2005/06/28ver:$

Thus, suitable a cannot be found, and therefore A-B ineq. is not satisfied.

14 Conclusion

References

A. Beznosikov, S. Horváth, P. Richtárik, , and M. Safaryan. On biased compression for distributed learning. *preprint arXiv:2002.12410*, 2020.

R. Szlendak, A. Tyurin, and P. Richtárik. Permutation compressors for provably faster distributed nonconvex optimization. *preprint arXiv:2110.03300*, 2021.

E. Gorbunov, K. Burlachenko, and P. Richtárik. MARINA: Faster Non-Convex Distributed Learning with Compression. *preprint arXiv:2102.07845*, 2022.

P. Richtárik, I. Sokolov, and I. Fatkhullin. EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback. *preprint arXiv:2106.05203*, 2021.

I. Fatkhullin, I. Sokolov, E. Gorbunov, Z. Li, and P. Richtárik. EF21 with bells whistles: practical algorithmic extensions of modern error feedback. 2021.

E. Chan, M. Heimlich, A. Purkayastha, and R. Geijn. Collective communication: theory, practice, and experience. 2007.

S. Li and T. Hoefler. Near-Optimal Sparse Allreduce for Distributed Deep Learning. *preprint arxiv:2201.07598*, 2022.

Chen Yu Tong Zhang Hanlin Tang, Xiangru Lian and Ji Liu. DOUBLESQUEEZE: Parallel Stochastic Gradient Descent with Double-pass Error-Compensated Compression. 2020.

Jiliang Tang Xiaorui Liu, Yao Li and Ming Yan. A Double Residual Compression Algorithm for Efficient Distributed Learning. 2019.

S. Shi, Q. Wang, K. Zhao, Z. Tang, Y. Wang, X. Huang, and X. Chu. A Distributed Synchronous SGD Algorithm with Global Top- k Sparsification for Low Bandwidth Networks. 2019.

Cedric Renggli, ETH Zurich Saleh Ashkboos, IST Austria Mehdi Aghagolzadeh, Microsoft Dan Alistarh, IST Austria Torsten Hoefler, and ETH Zurich. SparCML: High-Performance Sparse Communication for Machine Learning. 2019.

Dan Alistarh, Demjan Grubic, Jerry Z. Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. 2017.

Ka Chun Cheung Shaohui Shi, Xiaowen Chu and Simon See. Understanding Top-k sparsification in distributed deep learning. 2019.