Project ”Permutation Compressors” Report

Mmesomachi Nwachukwu [1] **Viacheslav Naumov** [1] **Alina Nurysheva** [1] **Anastasia Yagnych** [1]

Abstract

Distributed optimization methods play a crucial role in tackling problems related to large-scale ML models. The very basic aim is to reduce bottlenecks, reduce computational and communication overhead caused by complicated models in distributed systems. The MARINA method is the state-of-the-art and employs biased gradient estimators along with communication compression that minimize the number of communication rounds needed during convergence. This project report is about MARINA integrated with permutation compressors, or PermK, and RandK as a new way utilizing the correlation among workers' updates to improve communication efficiency. With an extensive consideration of theoretical foundations, practical manifestations, and experimental validations, we set out to compare and contrast the performance of MARINA against traditional methods. The results demonstrate how MARINA outperformed the competition in minimizing communication complexity and achieving faster convergence. Finally, this work highlights the challenging and yet meaningful subject of advanced compression techniques for distributed ML systems.

Github repository: Permutation Compressors.

1. Introduction

Optimization is the very foundation of ML, allowing one to train models spanning from simple regressions to complex deep neural networks. On top of this, distributed optimization has become an urgent necessity because of the great growth in size of most datasets and models. However, in distributed optimization, several workers collaborate to find a minimum of a common objective function, distributing computational load among nodes. However, such an approach introduces high communication overhead, especially related to the frequent exchange of gradient information between workers and a central server.

Traditional optimization algorithms like SGD(Beznosikov et al., 2023) broadcast the full gradient information during each iteration. Although they are very effective, their communication cost is substantially high for high-dimensional models. With regard to this challenge, some researchers develop some techniques for compressing gradients. These techniques reduce data communication during training and allow efficient communication without compromise on convergence.

Some popular ones include RandK and TopK sparsifiers; the former randomly selects a subset of gradient components, while the latter sends the components of the largest magnitude. These techniques are usually quite successful in improving communication efficiency, but suffer from losses in convergence speed. Normally, a trade-off is made in convergence speed to achieve a better communication complexity because loss of information by compression might decrease convergence.

The method of MARINA(Szlendak et al., 2021), introduced by Gorbunov et al., is a great step forward in distributed optimization. The combination of biased stochastic gradient estimators with communication compression in MARINA reduces the number of communication rounds required for convergence. One of the key novelties of MARINA is the use of permutation compressors (PermK), which was suggested by Szlendak et al. Unlike previous compressors, PermK exploits correlations among updates of workers and reduces communication variance, thus being more efficient.

This report covers the theoretical grounds of MARINA and PermK, their practical implementation, and performance in real-world scenarios. The main research question to be answered would then be: How do permutation compressors improve the efficiency of the distributed optimization methods at play, such as MARINA? By combining theoretical insights with experimental results, this study points out the transformative potential of MARINA combined with PermK for distributed machine learning.

2. Related work

Distributed optimization has been a central focus in machine learning research, driven by the need to scale algorithms across large datasets and computational resources. Traditional approaches, such as SGD, have long been the standard for optimization due to their simplicity and effectiveness.

However, reliance on full gradient transmission in SGD presents scalability challenges in distributed settings, where communication becomes a bottleneck.

Gradient compression techniques have emerged as a promising solution to address these challenges. Among the most popular methods are RandK and TopK sparsifiers. RandK randomly selects a subset of gradient components for transmission, while TopK focuses on transmitting the largest components based on their magnitude. These methods significantly reduce communication costs but often introduce biases or slow convergence due to the loss of gradient information. To mitigate these effects, error feedback mechanisms, such as those employed in EF21, have been developed. These mechanisms compensate for omitted components in subsequent iterations, partially restoring convergence speed.

MARINA builds upon these foundational techniques by introducing biased gradient estimators that reduce communication rounds while maintaining robust convergence guarantees. A notable extension of MARINA, proposed by Szlendak et al., incorporates permutation compressors (PermK). PermK leverages correlations among workers' updates, reducing communication variance and improving theoretical communication complexity. This approach is particularly effective in scenarios with low Hessian variance, where traditional methods often struggle to achieve optimal performance.

Empirical studies have demonstrated the practical benefits of MARINA and PermK in various machine learning tasks. For instance, experiments on synthetic datasets and real-world applications, such as MNIST classification, reveal that MARINA with PermK achieves faster convergence and lower communication costs compared to RandK and TopK. These findings underscore the importance of exploring advanced compression techniques to address the evolving challenges of distributed optimization.

This report extends these insights by providing a detailed analysis of MARINA with PermK. Through theoretical exploration, practical implementation, and experimental validation, we aim to highlight the potential of this method to advance the field of distributed machine learning.

3. Methodology

So, what is a PermK? For $d \geq n$ assume $d = qn$. Let $(\pi_1, \pi_2, ..., \pi_d)$ be a random permutation of (1, 2, ..., d) then for $i \in 1, ..., n$

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

for $d < n$ and $n = qn$ let $(\pi_1, \pi_2, ..., \pi_d)$ be a random permutation of (1, .., 1, 2,..., 2, ..., d, ..., d) where each appers q times, then

$$C_i(x) := dx_{pi_j} e_{\pi_j}$$

The CGD method can be performed as:

$$x_{k+1} = x_k - \eta C_k(\nabla f(x_k)),$$

where x_k - parameter vector at iteration k, $\eta > 0$ - step size, C_k - compression operator, $\nabla f(x_k)$ - gradient of the objective function f.

All in all, there are classes of compressing operators:

1. Class $B_1(\alpha, \beta)$:

$$\alpha||x||^2 \leq E[||C(x)^2||] \leq \beta\langle E[C(x)], x\rangle$$

2. Class $B_2(\gamma, \beta)$:

$$\max(\gamma||x||^2, \frac{1}{\beta}E[||C(x)||^2]) \leq \langle E[C(x)], x\rangle$$

3. Class $B_3(\delta)$:

$$E[||C(x) - x^2||] \leq \left(1 - \frac{1}{\delta}\right)||x||^2$$

As for complexities of the algorithms we have:

1. For $C \in B_1(\alpha, \beta)$:

$$E_k \leq \left(1 - \frac{\alpha\mu}{\beta^2 L}\right)^k E_0$$

2. For $C \in B_2(\gamma, \beta)$:

$$E_k \leq \left(1 - \frac{\gamma\mu}{\beta L}\right)^k E_0$$

3. For $C \in B_3(\delta)$:

$$E_k \leq \left(1 - \frac{\mu}{\delta L}\right)^k E_0$$

The MARINA method optimizes a nonconvex objective $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$, where $f_i(x)$ represents the loss on data stored at worker i, n is the number of workers. The update rule is given by: $x^{k+1} = x^k - \gamma g^k, \quad g^k = \dfrac{1}{n}\displaystyle\sum_{i=1}^n g_i^k,$

$$g_i^{k+1} = \begin{cases} \nabla f_i(x^{k+1}) & \text{if} \quad \theta_k = 1 \\ g^k + \mathcal{C}_i^k(\nabla f_i(x^{k+1}) - \nabla f_i(x^k)) & \text{if} \quad \theta_k = 0 \end{cases}$$

where C_i are compression operators. Permutation compressors (PermK) are designed to reduce communication variance while maintaining unbiased gradient estimates.

Marina algorithm:

Input: starting point x^0, stepsize γ, probability $p \in (0, 1]$, number of iterations T

Initialize $g^0 = \nabla f(x^0)$

for k= 0, 1, . . . , T- 1 **do**

Sample θ_t ∼Be(p)

Broadcast g^t to all workers

for $i = 1, \ldots, n$ in parallel do

$x^{t+1} = x^t - \gamma g^t$

Set $g_i^{t+1} = \nabla f_i(x^{t+1})$ if $\theta_t = 1$, and $g_i^{t+1} = g^t + \mathcal{C}_i\left(\nabla f_i(x^{t+1}) - \nabla f_i(x^t)\right)$ otherwise

end for

$g^{t+1} = \frac{1}{n}\sum_{i=1}^n g_i^{t+1}$

end for

Output: $\hat{x}^T$ chosen uniformly at random from $\{x^t\}_{k=0}^{T-1}$

Communication complexity is

$$T = O\left(\frac{\Delta_0}{\epsilon}\left(L_- + L_+\sqrt{\frac{1-p}{p}\cdot\frac{w}{n}}\right)\right),$$

where $\Delta_0 = f(x_0) - f^*$, $L_\pm$ are gradient smoothness constants.

The implementation involved:

- Setting up a distributed environment with multiple workers communicating with a central server
- Implementing MARINA with both RandK and PermK compressors
- Comparing performance across linear regression, logistic regression

3.1. Code Implementation

The following code snippet illustrates the implementation of MARINA with PermK compressors:

Listing 1. MARINA Implementation

```
def marina_update(x, gradients, compressor,
     step_size):
    compressed_gradients = [compressor(g)
        for g in gradients]
    avg_gradient = sum(compressed_gradients
        ) / len(compressed_gradients)
    return x - step_size * avg_gradient
```

4. Experiment description

Experiments were conducted to evaluate MARINA's performance with PermK compressors. The experimental setup included:

- **Tasks:** Linear regression, logistic regression
- **Compressors:** RandK and PermK were implemented and compared
- **Metrics:** Number of communication rounds, bits transmitted, and convergence rate

4.1. Results

For the data we used the one from homework "mushrooms.txt". Train part was 80%.

Figures illustrate the performance of MARINA with RandK PermK compressors within logistic regression. As shown, PermK consistently outperforms RandK in terms of convergence speed and communication efficiency.

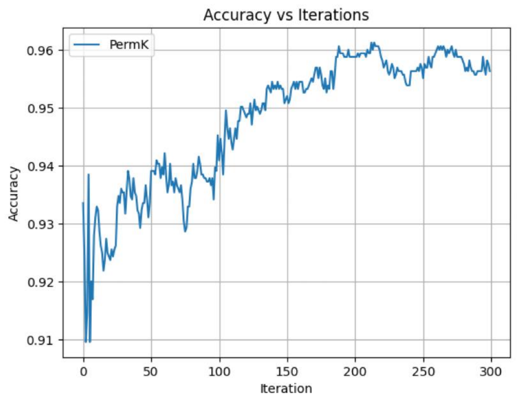

Figure 1. Loss vs. iterations for logistic regression for PermK

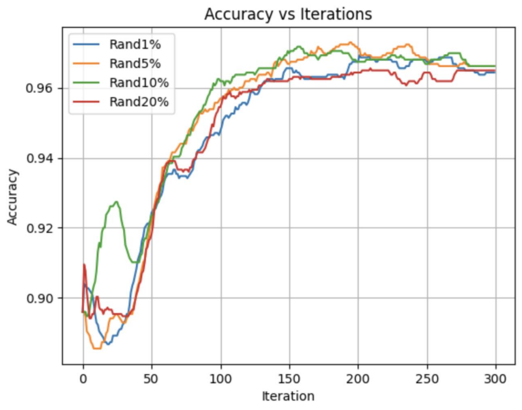

Figure 2. Loss vs. iterations for logistic regression for RandK

Was used the Ray module to implement the distributed structure and we used 300 iterations to look at the convergence and accuracy behavior.

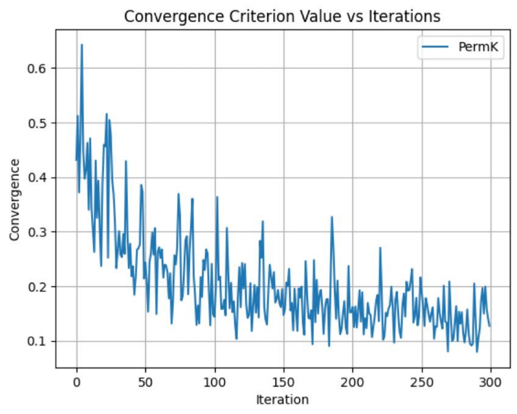

Figure 3. Convergence vs. iterations for logistic regression for PermK

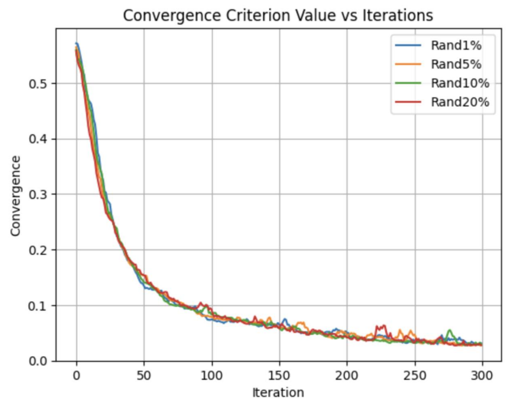

Figure 4. Convergence vs. iterations for logistic regression for RandK

Also we tried with different numbers of workers n and different levels of compression for linear regression:

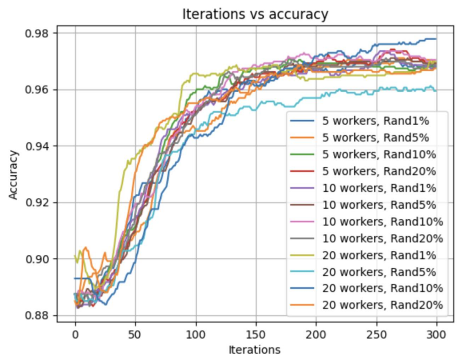

Figure 5. Accuracy vs. iterations for linear regression with different n of workers

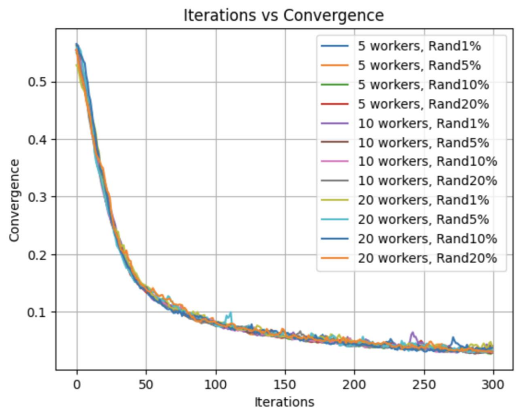

Figure 6. Convergence vs. iterations for linear regression with different n of workers

Also, we tried to tune the parameter p in Marina, the results are:

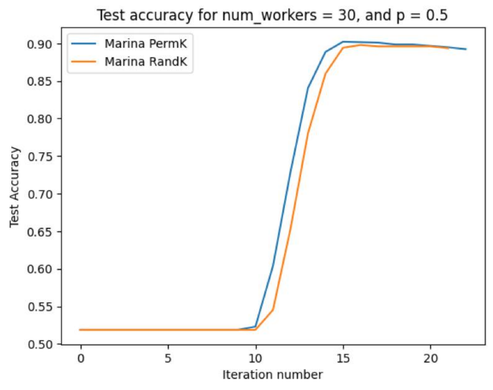

Figure 7. Accuracy for n=30, p=0.5

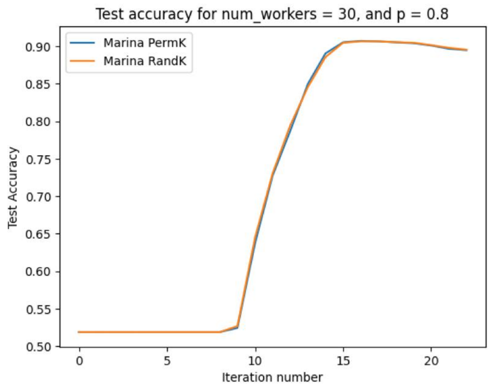

Figure 8. Accuracy for n=30, p=0.8

Finally, computed the RandK and PermK for CGD.

Was used the Ray module to implement the distributed structure. Default for 100 iterations and epsilon accuracy of 1e-8, but we noticed that convergence happened quite early (usually before 30th iteration).

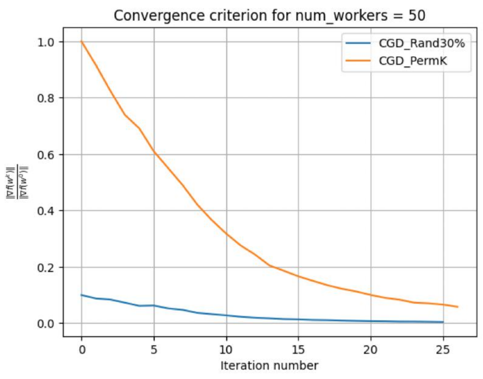

Figure 9. Convergence of CGD RandK vs PermK

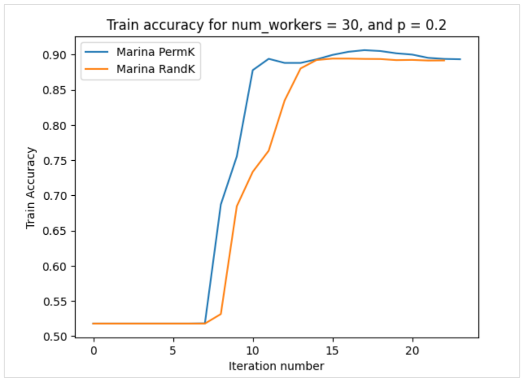

Figure 10. Accuracy of CGD RandK vs PermK

5. Discussion and conclusion

The experimental results validate the theoretical advantages of MARINA with permutation compressors. Key findings are:

- **Reduced Communication Complexity**. PermK significantly reduces the number of bits transmitted per iteration, making it ideal for large-scale distributed systems
- **Improved Convergence**. The correlation introduced by PermK enhances gradient estimates, leading to faster convergence compared to RandK
- **Scalability**. PermK's efficiency scales well with the number of workers, particularly in scenarios with low Hessian variance

References

Beznosikov, A., Horvth, S., Richtrik, P., and Safaryan, M. On biased compression for distributed learning. *Journal of Machine Learning Research*, 24(276):1–50, 2023. URL `http://jmlr.org/papers/v24/21-1548.html`.

Szlendak, R., Tyurin, A., and Richtárik, P. Permutation compressors for provably faster distributed nonconvex optimization. *CoRR*, abs/2110.03300, 2021. URL `https://arxiv.org/abs/2110.03300`.

A. Team member's roles and contributions

- Mmesomachi Nwachukwu (25% of work) - implemented CGD with RandK and PermK compressors
- Viacheslav Naumov (25% of work) - applied Marina method and logistic regression on it
- Alina Nurysheva (25% of work) - project presentation
- Anastasia Yagnych (25% of work) - applied logistic regression and linear regression for Marina, report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

The following reproducibility checklist will be done by the final version of the report.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment:

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment:

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment:

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment:

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment:

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.

- ☐ Not applicable.

Students' comment:

7. An explanation of how samples were allocated for training, validation and testing is included in the report.
   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment:

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.
   - ☐ Yes.
   - ☐ No.
   - ☑ Not applicable.

   Students' comment:

9. The exact number of evaluation runs is included.
   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment:

10. A description of how experiments have been conducted is included.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    Students' comment:

11. A clear definition of the specific measure or statistics used to report results is included in the report.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    Students' comment:

12. A description of the computing infrastructure used is included in the report.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    Students' comment: