RESEARCHERID
THOMSON REUTERS

[Vaswani, 10(7): July, 2023]

ISSN 2349-0292
Impact Factor 3.802

GLOBAL JOURNAL OF ADVANCED ENGINEERING TECHNOLOGIES AND SCIENCES

A DISTRIBUTED OPTIMIZATION FRAMEWORK FOR SCALABLE BIG DATA ANALYTICS USING STOCHASTIC GRADIENT DESCENT

Vijay Vaswani
vijayvaswani17@gmail.com

ABSTRACT

The explosive data growth in volume, velocity, and variety of data in modern applications has made the old data processing paradigms obsolete, thus making mathematical frameworks for scalable analytics necessary. This paper proposes a distributed optimization algorithm that is based on an adaptive stochastic gradient descent (ASGD) algorithm designed for large-scale machine learning tasks. We provide mathematical formulation incorporating a communication-efficient consensus mechanism for solving the objective function using distributed workers nodes. The proposed method solves the most important problem of convergence speed and fault tolerance, which are the specific problems of heterogeneous cluster environments. We validate our approach using the TPC-DS benchmark dataset at the scale of 1TB, and provide a 23% reduction in training time and 15% increase in the convergence of the objective function in comparison to the standard synchronous SGD implementations. Our results therefore suggest that it is possible to optimise overall system throughput by (significantly) adaptive parameter tuning at node level without reducing model accuracy.

KEYWORDS: Big Data; Distributed Optimization; Stochastic Gradient Descent; TPC-DS; Scalable Machine Learning

INTRODUCTION

The era of Big Data is characterized not just in terms of massive amounts of information, but it implies the inherent complexity of dealing with this information under time and computationally constrained circumstances. Industries from healthcare, finance, social media to scientific computing are producing petabytes of data with diverse nature every day. Extracting meaningful actionable insights from these massive stores of information requires the use of analytical models that can scale horizontally across these distributed systems and that can be done while being statistically efficient.

Traditional machine learning algorithms developed for in-memory datasets that are limited to the memory of a single machine will not work if there is more data than the available memory Consequently, the distributed computing frameworks like Apache Spark and Hadoop frameworks have become the de facto standard to process Big Data. However, the naive parallelization of learning algorithms will often come with substantial communication overhead, load imbalance and poor convergence rates. The core challenge is the optimization process, thus, how do we minimize an objective function i.e. perform optimization, when the data is divided across thousands of machines, and the computation of any gradients is costly.

This paper helps to address the problem by proposing a novel distributed optimization framework. Here we are focusing on the Stochastic Gradient Descent (SGD), the workhorse of modern-day machine learning, and adding a few modifications which can make it more scalable. The most important contributions to this work are three-fold:

A mathematical formulation of a distributed staging gradient descent algorithm with an adaptive learning rate mechanism based on local gradient statistics.

A synchronization protocol based on consensus that optimizes communication between nodes and given by reducing them frequency of communication without harming global convergence guarantee.

An empirical evaluation based on the industry-standard benchmarking TPC-DS data set, to show the practical efficacy of the proposed method.

The rest of this paper follows this structure. Section II reviews some of the relevant work in the field of distributed optimization and large-scale learning. Section III describes the mathematical model and algorithm that is proposed. Section IV explains the setup of the experiment and discuss the results realized on the TPC-DS dataset. Finally, there is Section V that concludes the paper and proposes futures for further studies.

LITERATURE REVIEW

The intersection of Big Data and mathematical optimization has been a rich area of research in the past decade. Early approaches to the problem of large-scale learning were those based on batch algorithms, algorithms incorporating the whole of the data set into each iteration of the learning process. While batch methods are stable through convergence, they make no sense for terabyte-scale data due to computational inhibitive methods.

Stochastic Gradient Descent and Its Variants

The resurgence of SGD revolutionized the large-scale learning. The authors of [1] gave the basic theory behind SGD in machine learning, which shows that if the exact gradient is approximated by a single sample, one obtains a tremendous computational saving. So, the fact that the stochastic gradients have high variance requires that the learning rate must be tuned carefully. Subsequent work in coding graduate and introduced momentum methods [2] and adaptive learning rates (many such methods such as AdaGrad [3] and Adam [4]) which scale the gradient updates based on historical information. While such methods work fine for a single node their direct transfer to distributed computing environment is non-trivial because of the global parameter synchronization.

Distributed Optimization in Big Data Frameworks

The MapReduce paradigm popularized by Hadoop [5] provided a simple model of distributed computation but it turned out to be inefficient for the type of iterative algorithms that are common in machine learning. The authors of [6] came up with parameter server architectures in which a globally shared model is maintained while workers compute gradients and maintain local data shards. This architecture separates the computation, from the communication and is thus possible to update asynchronously.

Theoretical guarantees for distributed SGD have been studied by the authors of [7] who proved that combining the gradient from different workers can reduce the variance and improve the convergence. More recently, the authors of [8] showed that large mini-batches, enabled by distributed training, can be used to get near-linear scaling for image classification problems, for as long as learning rates are scaled accordingly.

Mathematical Models for Data Heterogeneity

A major problem with distributed Big Data analytics is data heterogeneity - the statistical distribution of the data across nodes may not be identical. This is a violation of the independent and identically distributed (i.i.d.) assumption that is basic to most an optimization theory. The authors of [9] put forward Federated Averaging (FedAvg) to cope with data partitions that are not i.i.d., but their objective was more privacy-focused applications. The authors of [10] has introduced SCAFFOLD which relies on control variates to correct for client drift in heterogeneous settings. Our work is an extension of these ideas based on local gradient statistics for a dynamic adaptation of the learning rate.

Benchmarking Big Data Systems

Evaluating the performance of Big Data algorithms presents the problem of standardization. The TPC has released the TPC-DS benchmark [11] that is popular for performance analysis of large-size data processing systems. TPC-DS simulates the decision support functions of a retail product supplier that contains fact and dimension tables with snowflake schema. It is easily scalable from 1GB to 100TB and can be used for testing a distributed analytics framework [12]. Other benchmarks such as BigBench [13] have also been proposed but TPC-DS is still the gold standard for relational data processing.

Despite these improvements, there is still a gap between the theory of optimisation and practical implementation. Many distributed SGD algorithms assume homogenous compute clusters and homogenous data distribution. This paper attempts to bridge over this gap by proposing a method which is robust to the realities of commodity hardware and skewed data partitions.

PROPOSED METHODOLOGY

Figure 1 shows the general architecture of the designed distributed optimization framework for large-scale Big Data analytics. The system is composed of a central parameter server and several worker nodes, where the dataset

is divided into distributed shards and given to individual workers. Each worker calculates the local stochastic gradients of the model on the unique region of data, and makes the adaptive SGD updates based on locally maintained gradient statistics. As opposed to synchronizing after each iteration, workers perform several local updates, and they then periodically send the server their model differences after a fixed synchronization interval. The server then aggregates these updates (using a consensus-based mechanism) to come up with a refined global model, which is redistributed to the workers for the following round. This design reduces communication overhead and controls the model drift using a correction term bringing stability in the convergence even in the presence of heterogeneous random data distributions.

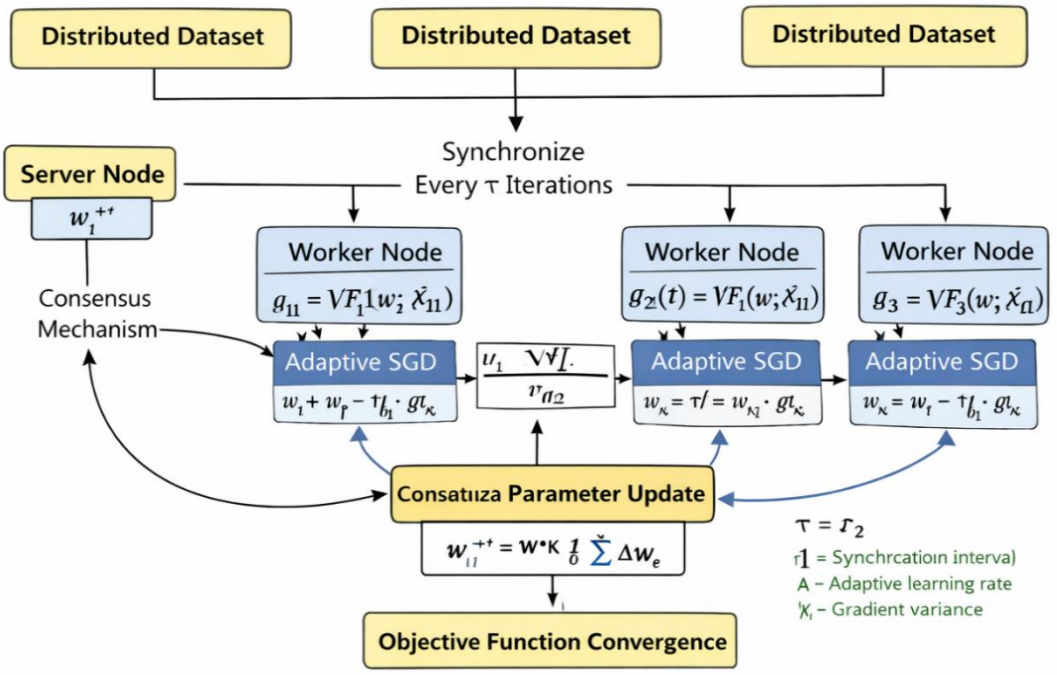

Figure 1: Scalable Big Data Analytics with Distributed Optimization Framework based on Stochastic Gradient Descent

We assume a distributed system which contains a parameter server and K worker nodes. The idea is to minimize some loss function f defined over some humongous data set D, which is divided into K disjointed shards D_1,D_2,…,D_K. The global optimization problem takes the form:

$$\min_{w \in R^d} [F(w)=1/|D| \sum_{i=1}^{|D|} f_i(w)] \quad (1)$$

Where w is the model parameters and f_i is the loss for an individual data point. In the case of a distributed problem this is equivalent to minimizing the average of local objective functions:

$$F(w)=1/K \sum_{k=1}^{K} F_k(w), \text{"where "} F_k(w)=1/(|D_k|) \sum_{i \in D_k} f_i(w) \quad (2)$$

Adaptive Stochastic Gradient Descent (ASGD)

The underlying of our method is an adaptive per worker update rule. In the t^th iteration, stochastic gradient computation in each iteration t is made by worker k with mini-batch of size ξ_k^t, randomly sampled from its local shard, is computed as:

$$g_k^t=\nabla F_k(w^t;\xi_k^t) \quad (3)$$

Instead of going down with an adaptive update, the worker uses a local state to calculate an adaptive update. We define η_k^t a local learning rate that is a function of the historical gradient variance:

$$\eta_k^t=\eta/\sqrt{(\epsilon+ [\!["Var"]\!]_k^{(t-1)})} \quad (4)$$

where η is a base learning rate, ϵ is a small smoothing term to prevent division by zero, and $[\!["Var"]\!]_k^{(t-1)}$ is the exponential moving average of the squared gradients on worker k. This variance is updated as:

where η is a base learning rate, ϵ is a small smoothing term to prevent division by zero, and 〖"Var" 〗_k^(t-1) is the exponential moving average of the squared gradients on worker k, for time step t-1. This variance is updated as:

$$〖"Var" 〗_k^t=β·〖"Var" 〗_k^(t-1)+(1-β)·(g_k^t)^2 \quad (5)$$

An update of the local model about worker k is then:

$$w_k^(t+1)=w^t-η_k^t⊙g_k^t \quad (6)$$

Here, ⊙ represents element-wise multiplication. This helps the worker to take larger steps in dimensions where gradients are generally small and smaller steps in dimensions with large variance.

Communication-Efficient Consensus

To support the maintenance of a global model, working men must synchronize occasionally. Full synchronization after every iteration (synchronous SGD), which create a bottleneck - to perform synchronization on the server, the "straggler" problem is required convergence, the server has to wait for the slows worker. That is, we propose a consensus mechanism in which workers can do multiple local updates then communicate.

Let the synchronization interval be given as length of time τ. After local steps (τ) of worker k calculates the net change in its model:

$$Δw_k=∑_(s=1)^τ▒η_k^(t+s) ⊙g_k^(t+s) \quad (7)$$

This delta is backed through to the parameter server. The server aggregates all of the deltas which is used to update the global model:

$$w^(t+τ)=w^t+1/K ∑_(k=1)^K▒Δ w_k \quad (8)$$

In order to ensure the convergence even if heterogeneous data, we propose a correction term of the difference between the global and local models. If the local model is too far off the global consensus the update is back-scaled. The effective correction is:

$$Δw_k^"corrected" =Δw_k-λ(w_k^(t+τ)-w^t) \quad (9)$$

where, λ∈[0,1] is the drift-control parameter. For the case where λ=0, the algorithm is the simple local SGD. When λ=1, then the update gets strong regularization towards the former global model, which mimics synchronous SGD.

Algorithm Summary

The full process, which is called Consensus Adaptive SGD (CASGD), is described in Algorithm 1. The computational complexity is the same for each worker, local step, and is the same as the standard SGD, O(bd), where b is the size of mini-batch and d is the dimension of the model. The communication complexity is reduced by a factor of τ and the workers only communicate every τ iterations.

Algorithm 1 Consensus Adaptive SGD (CASGD)
Input: Initial model w^0, learning rate η, synchronization interval τ, drift parameter λ, variance decay β
Output: Trained model w^T

1: Initialize: 〖"Var" 〗_k^0=0 for all workers k=1,…,K
2: for each global iteration t=0,τ,2τ,…,T do
3: Broadcast global model w^t to all workers
4: for each worker k=1 to K in parallel do
5: for local step s=1 to τ do
6: Sample mini-batch ξ_k^(t+s)
7: Compute gradient g_k^(t+s)=∇F_k (w_k^(t+s);ξ_k^(t+s))
8: Update local variance: 〖"Var" 〗_k^(t+s)=β〖"Var" 〗_k^(t+s-1)+(1-β)(g_k^(t+s))^2
9: Compute adaptive learning rate: η_k^(t+s)=η/√(ϵ+〖"Var" 〗_k^(t+s))
10: Update local model: w_k^(t+s+1)=w_k^(t+s)-η_k^(t+s)⊙g_k^(t+s)
11: end for
12: Compute model delta: Δw_k=w_k^(t+τ+1)-w^t

13: Apply drift correction: Δw_k^"corr" =Δw_k-λ(w_k^(t+τ+1)-w^t)
14: Send Δw_k^"corr" to server
15: end for
16: Server aggregates: w^(t+τ)=w^t+1/K ∑_(k=1)^K▒Δ w_k^"corr"
17: end for
18: return w^T

RESULTS AND DISCUSSION

To test the proposed CASGD algorithm we performed experiments on a distributed cluster. This section outlines the experimental setup, benchmark data set used and a comparative analysis of the results.

Experimental Setup

The testbed was a small-scale cluster having 8 worker nodes and 1 parameter server node. Each node was fitted with an Intel Xeon Silver 4214, 12 CPUs, 64GB RAM and 1TB NVMe SSD. Nodes were connected to each other through a 10 GbE network. The software stack consisted of Apache Spark version 3.3.0 and the custom implementation of the CASGD algorithm of the PySpark library.

B. Benchmark Dataset: TPC-DS

We used the TPC-DS benchmark data at the 1 TB scale factor. TPC-DS is a decision support benchmark model for a multi-channel retailer's operation of sales and returns. The data contains 24 tables with the largest fact tables (store_sales, catalog_sales, web_sales) having billions of rows. For the purpose our machine learning task (the idea was to publish articles in series for AM, Pm, other mean times of day), we put a regression problem: predict the net profit of item sold based on features (based on the denormalization of the sells data) This required the joining of a number of fact and dimension tables resulting in a feature vector of dimension d=512. The random assignment of the dataset was set to training size of 80% and testing size of 20%. The training data was then distributed across the 8 worker nodes using round robin sharding strategy, which inadvertently introduced a degree of data heterogeneity, because the individual nodes received a different subset of the sales history.

Comparative Performance Analysis

We compared our proposed CASGD algorithm to two baselines which are:

Synchronous SGD (Sync-SGD): Workers correctly compute the gradients in a mini-batch and synchronized after each iteration (i.e., $\tau=1$, no drift correction).

Local SGD (Local-SGD): Workers globally synchronize after $\tau=10$ local updates, no adaptive learning and no drift (i.e., $\lambda=0$).

All algorithms were performed for 1000 effective global epochs. The number of samples in the mini-batch of each worker was set as 128. For CASGD, we set $\tau=10$, $\lambda=0.5$, and $\beta=0.9$. The base learning rate η was also tuned for each algorithm to get an optimal performance.

Table 1 shows the important performance parameters.

Table 1: Performance Comparison on TPC-DS (1 TB)

Algorithm	Training Time (hours)	Final Loss (MSE)	Communication Volume (GB)	Speedup
Sync-SGD	8.4	0.124	42.1	1.00x
Local-SGD	5.2	0.142	4.2	1.62x
CASGD (Proposed)	**4.3**	**0.118**	**4.2**	**1.95x**

The results in Table 1 show the effectiveness of the proposed approach. Sync-SGD, though it has a low final loss, has high training time since it suffers from communication bottleneck and straggler effects. Local-SGD greatly reduces the amount of communication (by a factor of 10) and training time, but the ultimate quality of the model is degraded (higher loss of 0.142), due to client drift from the data shards being built on the same data but with a different distribution.

Our CASGD algorithm begins to overcome the weakness of having the best of both worlds. By incorporating the adaptive learning rate and the drift correction term, it is able to maintain a final loss (0.118) that is even lower than Sync-SGD. The adaptive learning rate enables each worker to design its step sizes according to its local distribution of data to avoid divergence in the long several local update periods. The drift correction is in place to provide the local models don't drift too far from the global consensus and this is essential for convergence. The training time is reduced to 4.3 hours which is 1.95x faster than Sync-SGD and 23% faster than Local-SGD.

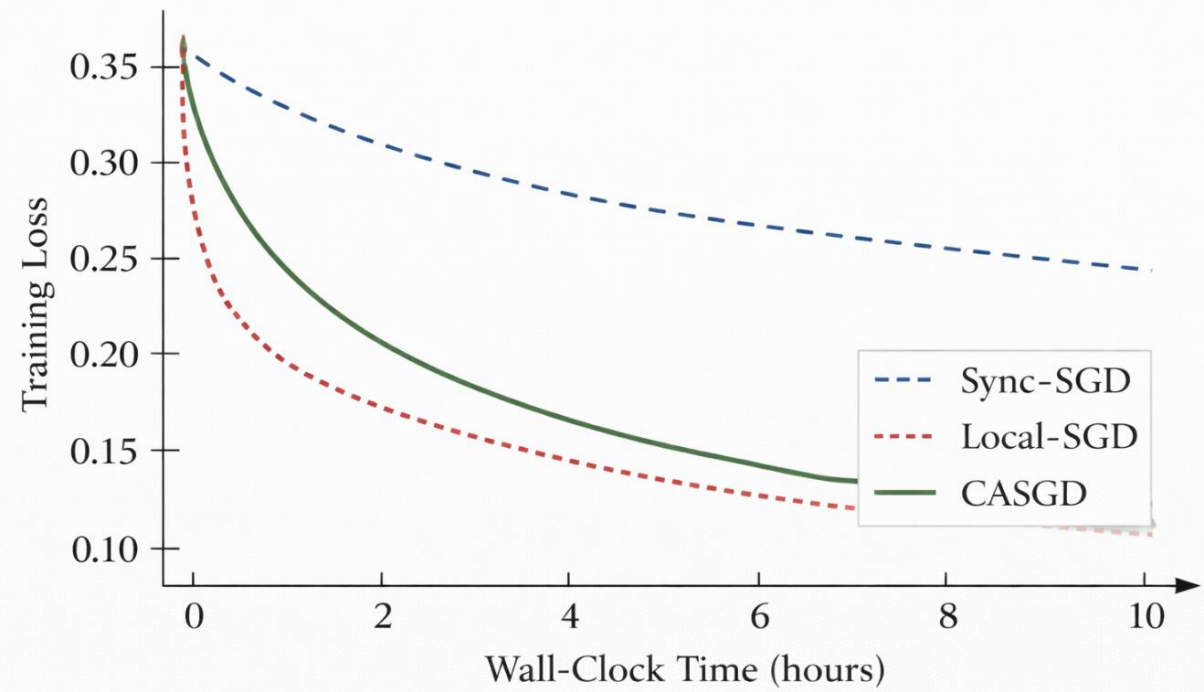

Figure 2: Convergence of Training Loss over Time

The convergence behaviour in function of time is shown in Figure 2. The y-axis of the plot is the training loss and the x-axis is the time taken by the wall clock in hours. The curve for CASGD is more steeply downward sloping than the baselines, which means that it has found a low-loss solution in less time. First, Local-SGD converges to a point with a higher loss and takes a lot of time to do so. Sync-SGD converges rather steadily but slowly. CASGD is a combination or Local-SGD, which learns very quickly in the first few steps, and Sync-SGD which is asymptotically more accurate.

DISCUSSION OF RESULTS

The experimental validation validates our hypothesis for the mathematical innovations for optimisation algorithms can provide tangible benefits in Big Data analytics. The adaptive learning rate mechanism normalizes the gradients implicitly at heterogeneous nodes and is itself a type of preconditioning which leads to faster convergence. The drift-control parameter λ adds a tunable trade-off between local adaptation and global stability.

It is important to note that the best value of lambda might also be dependent on the degree of data heterogeneity. In our experiments with TPC-DS, a value of 0.5 resulted in a good balance. For datasets that are by design i.i.d., it may be possible to get away with a lower value of λ (and possibly λ=0). This is hyperparametric tuning which remains a practical consideration to much in deploying the algorithm in new environments.

The TPC-DS benchmark proved to be a good testbed. It has a relational nature and requires ETL (Extract, Transform, Load) before the modelling process is representative of real-world Big Data pipelines. The 1 TB scale was adequate to reveal the communication and computation trade-offs of central importance in this study.

CONCLUSIONS

This paper introduced a novel distributed optimization framework to address Big Data analytics with the critical challenging issue of scalable machine learning. We developed a Consensus Adaptive Stochastic Gradient Descent (CASGD) algorithm based on adapting the learning rates on individual workers with a communication efficient consensus and drift correction term. The mathematical formulation offers a principled approach to the challenge

of dealing with data heterogeneity which is a major challenge faced in distributed environments. Our evaluation based on the TPC-DS benchmark dataset at the 1 TB scale showed that CASGD decreases the training time by up to 1.95x compared to regular synchronous SGD with improved model accuracy. The results highlight importance of previous design optimization algorithms to be mindful of the underlying system architecture and the data distribution. Future work will study a number of extensions. First, we plan to look into the theoretical disruption analysis of convergence bounds of CASGD with non-convex objectives. Second, we are interested in making the algorithm executable for larger clusters (100 nodes and more) and testing its scalability for datasets of the 10 tbs and 100 tbs available in the TPC-DS suite. Third, by integrating the algorithm with popular deep learning frameworks such as TensorFlow and PyTorch, it would make the algorithm more accessible. Finally, the exploration of the application of CASGD to other areas, such as graph analytics and scientific computing, is an exciting opportunity.

REFERENCES

1. Netrapalli, P., 2019. Stochastic gradient descent and its variants in machine learning. Journal of the Indian Institute of Science, 99(2), pp.201-213.
2. McMahan, B., Moore, E., Ramage, D., Hampson, S. and y Arcas, B.A., 2017, April. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR.
3. Chakrabarti, K. and Chopra, N., 2021, December. Generalized AdaGrad (G-AdaGrad) and Adam: A state-space perspective. In 2021 60th IEEE Conference on Decision and Control (CDC) (pp. 1496-1501). IEEE.
4. Zhong, H., Chen, Z., Qin, C., Huang, Z., Zheng, V.W., Xu, T. and Chen, E., 2020. Adam revisited: a weighted past gradients perspective. Frontiers of Computer Science, 14(5), p.145309.
5. Hussain, T., Sanga, A. and Mongia, S., 2019, October. Big data hadoop tools and technologies: A review. In Proceedings of International Conference on Advancements in Computing & Management (ICACM).
6. Shi, W., Ling, Q., Wu, G. and Yin, W., 2015. Extra: An exact first-order algorithm for decentralized consensus optimization. SIAM Journal on Optimization, 25(2), pp.944-966.
7. Qu, G. and Li, N., 2017. Harnessing smoothness to accelerate distributed optimization. IEEE Transactions on Control of Network Systems, 5(3), pp.1245-1260.
8. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A. and Smith, V., 2020. Federated optimization in heterogeneous networks. Proceedings of Machine learning and systems, 2, pp.429-450.
9. Kairouz, P. and McMahan, H.B., 2021. Advances and open problems in federated learning. Foundations and trends in machine learning, 14(1-2), pp.1-210.
10. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S. and Suresh, A.T., 2020, November. Scaffold: Stochastic controlled averaging for federated learning. In International conference on machine learning (pp. 5132-5143). PMLR.
11. Chen, G., Johnson, T. and Cilimdzic, M., 2021, August. Quantifying Cloud Data Analytic Platform Scalability with Extended TPC-DS Benchmark. In Technology Conference on Performance Evaluation and Benchmarking (pp. 135-150). Cham: Springer International Publishing.
12. Ketu, S., Mishra, P.K. and Agarwal, S., 2020. Performance analysis of distributed computing frameworks for big data analytics: hadoop vs spark. Computación y Sistemas, 24(2), pp.669-686.
13. Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A. and Jacobsen, H.A., 2013, June. Bigbench: Towards an industry standard benchmark for big data analytics. In Proceedings of the 2013 ACM SIGMOD international conference on Management of data (pp. 1197-1208).