Journal of Innovation in Applied Research

ISSN:2581-4206 (Online)
Available at http://jiar.in

Research Article

Implementing Fault Resilient Strategies in Cloud Computing via Federated Learning Approach

Lokendra Gour and *Akhilesh A. Waoo

Department of Computer Science and IT, Faculty of Computer Applications & Information Technology and Sciences, AKS University, Satna (M.P.) 485001

[1]Corresponding Author E-mail: akhileshwaoo@gmail.com

Received on: 11.01.21; Revised on: 21.01.21; Accepted on: 31.01.21

Abstract

Faults are inevitable in a very large scale distributed computing system such as cloud computing. The size of distributed computing is enlarging drastically due to the advent of the Internet of things (IoT). Faults occur frequently at any working node and cause the partial or complete failure of the cloud applications. Implementing fault resilient systems and securing cloud systems have become key challenging problems in recent years. A novel model with federated learning (FL) is analyzed and proposed to deal with these challenges. Federated learning, a special kind of distributed deep learning, works in collaboration with the distributed computing machines. A Federated learning model can be deployed on multiple clusters of computing nodes. One of the features of distributed computing is that it is growing drastically towards horizontal and vertical directions. The Federated learning model is deployed on both horizontal and vertical scaling. FL deployed with distributed deep learning can identify, recognize, and resolve the faults to great extent.

Keywords: Distributed Computing, Fault Resilience, Federated Learning, IoT, Cloud Computing

Introduction

To reduce the adverse effects of faults, machine learning (ML) especially the federated learning (FL) approach is deployed. Federated learning is a distributed and decentralized paradigm of protocols. The Federated learning approach is well suited for a distributed system because a set of worker machines (or nodes) can train the local models. Different chunks of datasets are distributed among the worker nodes or third parties. Here sections of a dataset are not shared by the working computational nodes. Thus federated learning is also the most significant model for achieving data privacy and data security in addition to fault tolerance. The existing FL approaches highlight optimizing only one dimension of the target space. The proposed methods can reduce communication costs and improve the efficiency of distributed computing. Federate deep learning (FDL) method minimizes the adverse effects with an improved convergence rate. This approach utilizes a weighted aggregation for accuracy improvement. FDL is capable to detect and diagnose the faults that occur frequently on end-user devices as well as on the edge. FDL is a novel communication efficient FL approach. It incorporates both synchronous and asynchronous arrangements.

Federated learning (FL) is a multi-modal machine learning system that trains the algorithm among various distributed and decentralized edge devices that holds local datasets. The intelligent device such as PDAs, smart-phones, and desktops or tablets system has been scaling rapidly in recent years. Most of these devices are equipped with multiple

sensors that allow them to produce and consume a huge amount of information. Distributed computing hierarchy consists of cloud, edge, and end-user devices. End-user devices train the local models and use local datasets.

End device and client's behavioral heterogeneity become the key cause of fault inclusion in cloud systems. The cloud system plays a major role in scaling big data.

Preliminaries

Federated learning models include hundreds of thousands of remotely distributed end devices. These devices use their device-generated data or section of datasets provided by the parameter server. All the participating devices get connected with a central server to obtain updated model parameters. In general, the principal objective of federated learning is typically to solve the following:

$$\min f(x) = \sum_{k=1}^{m} (p_k F_k(w)),$$

where m is the total number of participating end devices and $p_k \geq 0$. The end devices' objectives are to calculate the gradients of local models. The stochastic gradient descent (SGD) is most probably used algorithm in local devices. After aggregating the local gradients, federated learning generates the global model parameters for obtaining the final inference.

Methodologies

To implement a federated learning strategy, initially, a deep algorithm is deployed on each participating end device for the estimation of the local gradients of the loss function. FL models are deployed on clusters of end devices. FL collaborates and coordinates each end device or clusters of end devices with the help of parameter servers. The following algorithm establishes the correspondence between the server and the client processes.

Amazon SageMaker framework is used to implement the proposed FDL model. Following steps are carried out to accomplish the task.

Creating a notebook instance.

- Preparing the data for preprocessing.
- Training the proposed model with appropriate datasets.
- Deploying the model on designated cloud.
- Evaluating the proposed mode for measuring the performance.
- Monitoring the model's performance and accuracy.

A simple distributed algorithm

```
process p1
   var u IN init 0
   ACK: Boolean init true
   begin
      ~ACK ∧ REC (s) → ACK := true; u := u+1
       ACK           → send (t); ACK := false
   end
process p2
   var wait : Boolean init true
   begin
      ~Wait          → send (s); wait := true
      Wait ∧ REC (s) →Wait:= false
   end
```

Here p1 is the state of the process at the end device and p2 is the state of the process outside of the device i.e., at the cloud where the central servers are deployed.

The parameter or central server is deployed at the cloud layer to orchestrate and coordinate the local machines at the end devices. The parameter server performs the task of aggregating the local updates and upgrading the global model after receiving the updated local models. Edge works as an intermediate layer between the cloud and end-devices. Edge acts almost similar to the cloud, it performs the task of taking the output of end devices as input, applies aggregation and classification if necessary, and finally transfers its intermediate output to the cloud system for further processing if required. The participating end device uses local datasets and local models for training the local datasets and thus reduces the occurrence of faults to a great extent.

Parallel / Distributed Processing Mechanism

In a cloud system, multiple processes are running simultaneously on servers distributed or scattered across the globe. In parallel computing, a task or program is divided into multiple processes. Each process is executed by the processor of the single-processor or multi-processor system. Whenever multiple processes are executed simultaneously by the multi-processor system, it is known as parallel processing.

Distributed computing is an extension of parallel computing in the sense that parallel

processing is performed on the processing units distributed geographically. In case parallel processing is carried out in the cloud computing environment; the processing devices are distributed across the different locations most probably on servers of the data centers. Some protocols must be followed in parallel processing. For example, in a system program like UNIX/LINUX operating system environment a system routine known as the fork is called for creating a new instance of a process.

```
RetVal = fork ();
if ( RetVal == 0 )
{
   child ();
   {
      :  //child process
      :  //starts running
      :  //on end device
   }
}
else
{
   if ( RetVal == -1)
      Display (child process creation failed);
   else
   :  // parent/master process
   :  // starts running
   :  // at this point
}
```

Synchronization

A distributed system under the cloud environment suffers the practical problem of synchronization among the processes and heterogeneous resources. A very common Lock and Unlock mechanism are used in the proposed system to deal with the synchronization problem. Following is the code for the lock and unlock procedure:

```
Lock (L)
   {
   While (L == 1)
      {
         No Operation:
      }
Unlock (L)
   {
      L = 0; }
```

In this algorithm, L is the locking variable that works as an entry point. If L = 0, the entry is open and when L = 1, the entry is closed. When a process on the participating end device needs to access the shared data, it invokes a Lock (L) procedure. When the values of L becomes 0, Lock (L} procedure sets its value to 1. An Unlock (L) procedure makes the value of L to 0 (reset).

System Topology

System topology for federated with distributed learning consists of a graph G = (V, E), where V =set of working nodes (or sequential processes) E=set of edges (bi/unidirectional communication channel or links). Figure: 1.1 shows the graph of the proposed model. In the graph, nodes represent heterogeneous edge or end devices and the edge represents the communication channel. The links between the nodes may be guided on unguided. In the context of cloud computing nodes may also represent the groups or clusters of the end devices.

The proposed graph topology for the distributed federated learning approach deploys multiple edge (or end) devices like mobile phones, smart sensors, etc. The end device executes the process and the communication among the processes is performed through the message passing. Each edge device has its model with a local dataset.

The end device performs the task of calculation of local gradients of loss function based on localizing dataset. At the cloud end, these gradients are aggregated and updated for optimizing the model.

Experimental Setups

Python programming language is primarily used for building the model and for statistical analysis R language is deployed. Open source TensorFlow and PyCharm computational frameworks have been used for building ML and DL models. AWS platforms, Amazon SageMaker, and the AWS Deep Learning AMIs are implemented to build, train and deploy the proposed model.

Amazon SageMaker Studio has been implemented for development, training, deployment and monitoring the proposed FDL approach.

An open simulating platform, FLASH is also applied for designing and implementing the proposed FL model.

Contributed Datasets

Appropriate and relevant datasets are essential for achieving the accuracy of the system. Google AudioSet is a very large dataset which comprises human generated sound clips. It is deployed in the proposed model for training to investigate and compare miscellaneous audio signals under the noise parameter. Other datasets like ImageNet and MNIST are utilized for a comparative study of the model. ImageNet is a massive dataset of visual data. It is used in order to train the model to identify and classify the images. One of the synthetic datasets Reddit is also used for performance accuracy. Reddit is also used for text classification, model analysis and model predictions.

Two benchmark datasets CIFAR-10 and CIFAR-100 are implemented. They contain the large collection of colored tiny images of different categories like vehicles, animals, birds and miscellaneous things. The proposed FDL approach identifies and recognizes all these categories as such input data are available from the edge devices through the censors. Convolutional neural network (CNN) has been implied as it is the best method to object recognition work on CIFAR datasets. The training datasets are randomly bifurcated. 70% of the dataset has been used for training and the remaining 30% of the dataset has been used for testing and validation.

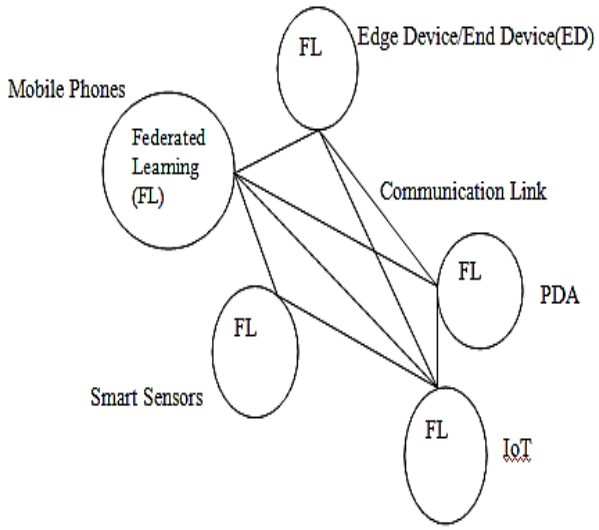

FIG. 1. A GRAPH OF EDGE DEVICES WITH DISTRIBUTED FEDERATED LEARNING

CONCLUSIONS AND FUTURE WORK

In the proposed model the participating edge device trains its model with a local dataset. These local datasets are not sharable among the edge devices hence the system preserves the privacy-sensitive personal data. Federated learning collaborates with machine learning without centralized training of the data.

Federated learning poses some of the key problems which have to be resolved: one of the problems is communication cost and the other one is the unreliability of the end devices that need not necessarily participate in the FL process. The proposed line of work opens the options for further research in direction of data security and privacy of personal data.

The existing FL systems are not preserving the heterogeneity of data and device heterogeneity. Heterogeneity reduces the convergence rate of FL. This is one of the core challenges in designing the FL model.

ACKNOWLEDGEMENTS

The authors would like to express special thanks to AKS University, Satna (MP) for providing a computer lab, infrastructure, technical personnel

and all the facilities to accomplish this research work. The authors would also like to thank all those who helped us a lot in finalizing this research project within the limited time frame.

CONFLICT OF INTEREST

The author declares no conflict of interest.

REFERENCE

1. Alistarh, D., Allen-Zhu, Z., Li, J. (2018): Byzantine stochastic gradient descent. In: Advances in Neural Information Processing Systems, 4613-4623.
2. Calheiros, R.N., Ranjan, R., De Rose, C.A.F., Buyya, R. (2009). CloudSim: A Novel Framework for Model and Simulation of Cloud Computing Infrastructures and Services, 1-9.
3. Kocher, D., Hilda, A.K.J. (2017). An approach for fault tolerance in cloud computing using a machine learning technique. Int. J. Pure Appl. Math. 117(22), 345-351.
4. Bekkerman, R., Bilenko, M., and Langford, J. (2011). Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.
5. Bernstein, J., Xiang Wang, Y., Azizzadenesheli, K., and Anandkumar, A. (2018). Signsgd: Compressed optimization for non-convex problems. In International Conference on Machine Learning, 559-568.
6. Bijral, A. S., Sarwate, Anand D., and Srebro N. (2016). On data dependence in distributed stochastic optimization. arXiv preprint arXiv:1603.04379.
7. Chaturapruek, S., John, C. D., and C. R´e, C. (2015). Asynchronous stochastic convex optimization: the noise is in the noise and sgd don't care. In Advances in Neural Information Processing Systems, 1531-1539.
8. Patil, A., Shah, A., Gaikwad, S., Mishra, A.a., Kohli, S.S., Dhage, S. (2011). Fault Tolerance in Cluster Computing System. In: 2011 Int. Conf. P2P, Parallel, Grid, Cloud Internet Compute, 408-412.
9. Hazan, E. Introduction to online convex optimization. Foundations and Trends in Optimization (2016). 2(3-4): 157–325.
10. He, K., Zhang X., Ren, S. and Jian S., Deep residual learning for image recognition. (2016). In Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778.
11. Engelmann, C., Vallée, G.R., Naughton, T., Scott, S.L. (2009). Proactive fault tolerance using preemptive migration. In: Proc. 17th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. PDP 2009, 252-257.
12. Kakade, S. M., Shwartz, S. S., and Tewari, A. (2012). Regularization techniques for learning with matrices. ournal of Machine Learning Research, 13(Jun), 1865-1890.
13. Shwartz, S. S., and David, S. B. (2014). Understanding machine learning: From theory to algorithms, Cambridge University Press.
14. Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In International Conference on Machine Learning, 928-936.
15. Hatcher, W. G., And Yua, W. (2018). Survey of Deep Learning: Platforms, Applications, and Emerging Research Trends, IEEE Access, May 24.
16. Chen X.W. and Lin X. (2016). Big data deep learning: Challenges and perspectives, IEEE Access, vol. 2, 2014. 14. Y. Ding, S. Chen, and J. Xu, Application of deep belief networks for opcode based, 514-525.
17. Malware detection, in Proc. Int. Joint Conf. Neural Netw. (IJCNN), 3901-3908.
18. Yuan, Y., and Jia K. (2015). A distributed anomaly detection method of operation energy consumption using smart meter data, in Proc. Int. Conf. Intell. Inf. Hiding Multimedia Signal Process. (IIH-MSP), 310-313.
19. Zhu, D., Jin, H., Y, Y., Wu, D., and Chen, W. (2017). DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data, in Proc. IEEE Symp. Comput. Commun. (ISCC), 438-443.
20. Ujjwalkarn. (2016). an Intuitive Explanation of Convolution Neural Networks. [Online]
21. Nielsen, M. (2018). Neural Networks and Deep Learning. [Online]. Available: 19. Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li.
22. Byzantine stochastic gradient descent. In Advances in Neural Information Processing Systems, 4613-4623.
23. Blanchard, P., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In Advances in Neural Information Processing Systems, 119-129.

24. Li, M., G., D., Andersen, Park, J. W., Smola, A. J., Ahmed, Josifovski, A., Long, J., J., I., Shekita, and Yiing B. Su. (2014). Scaling distributed machine learning with the parameter server. In Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, OSDI'14, Berkeley, CA, USA, 2014. USENIX Association, 583-598.
25. McMahan B. and Ramage, R. (2017). Federated learning: Collaborative machine learning without centralized training data. Google Research Blog, 3.
26. Eugene Bagdasaryan, Andreas Veit, YiqingHua, Deborah Estrin, and Vitaly Shmatikov. (2018). How to backdoor federated learning. arXivpreprint arXiv:1807.00459 (2018).
27. Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anand kumar. 2019. signSGD with Majority Vote is Communication Efficient and Fault Tolerant. In 7th International Conference on Learning Representations, ICLR (2019), New Orleans, LA, USA, May 6-9, 2019.
28. Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. (2019). Towards federated learning at scale: System design. arXiv preprint arXiv:1902.01046 (2019).
29. Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the (2017) ACM SIGSAC Conference on Computer and Communications Security. 1175–1191.
30. Daniel Ramage Brendan McMahan. (2017). Federated Learning: Collaborative Machine Learning without Centralized Training Data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. (2017). Published April 6, 2017.
31. Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečny, H Brendan McMa- han, Virginia Smith, and Ameet Talwalkar. (2018). Leaf: A benchmark for federated settings. arXiv preprint arXiv:1812.01097 (2018).