Challenges of Distributed Computing in the context of Deep Learning

Lokendra Gour, Akhilesh A. Waoo
Assistant Professor, Associate Professor and HOD
Department of Computer Science, Department of Computer Science Applications and Technology
RGCCAT Satna(M.P.) India, AKS University Satna(M.P.) India.

Abstract: In *recent years software development industry is looking forward a new trend of development of distributed software. Distributed software systems play a significant role where processing units or entities are scattered and interconnected over the computer networks. Efficiency of computer system is drastically improved by technological advancement in computer hardware systems. There is a great need of deep learning approach to cope with the problem managing massive resources in distributed system. Machine learning (ML) provides the mechanism to solve the critical problems occurred in distributed computing. Deep learning employs multiple processing elements or agents to each resource of distributed system. Data-parallelism is the basic approach for deep learning. Data security and privacy are the major issues to implement machine learning and deep learning methods. This paper is an extensive survey on challenges and problems of distributed system.*

Keywords*:* D**istributed computing, machine learning, deep learning, data parallelism, data security, privacy.**

I INTRODUCTION

In classical computing environment algorithms are deterministic and operational oriented. On the other hand machine learning algorithms are probabilistic, optimization oriented and iterative-convergent in nature. A serious problem with traditional computing is that it must follow the strict accuracy and correctness in the program. Single error in the algorithm leads to system failure. There is no provision of tolerating faults in any cost. In distributed computing data security and data privacy are the extremely necessary and demanding needs.

Machine learning computations have the following characteristics:

- Error Tolerance: ML algorithms are capable to handle limited number of errors occurred in intermediate calculations. Regardless of the few errors system will continue to provide an optimum solution
- Dynamic structural dependency: Parameters or variables are subject to change the co-relationships between them. To maintain the efficiency of the system, parallel structure of the program must change.
- Non-uniform convergence: Parameters of ML algorithms may converge in varying number of steps. Most of the parameters converge very quickly, while the remaining needs thousands of iterations.

II CHALLENGES AND PROBLEMS IN DISTRIBUTED COMPUTING

Distributed systems introduce many new problems that we might never have been forced to address in single systems:
In a single system it may be very easy to tell that one of the service processes has died (e.g. the process exited with a fatal signal or error return code). In a distributed system our only indication that a component has failed might be that we are no longer receiving messages from it. Perhaps it has failed, or perhaps it is only slow, or perhaps the network link has failed, or perhaps our own network interface has failed. Problems are much more difficult to diagnose in a distributed system, and if we incorrectly diagnose a problem we are likely to choose the wrong solution.
If we expect a distributed system to continue operating despite the failures of individual components, all of the components need to be made more robust (e.g. greater error checking, automatic fail-over, recovery and connection reestablishment). One particularly tricky part of recovery is how to handle situations where a failed component was holding resource locks. We must find some way of recognizing the problem and breaking the locks. And after we have broken the locks we need some way of (a) restoring the resource to a clean state and (b) preventing the previous owner from attempting to continue using the resource if he returns.

Following are the major challenges and problems of distributed computing:

- Byzantine general problem
- Heterogeneity
- Transparency
- Concurrency
- Security
- Scalability

- Failure Handling
- Quality of service
- Reliability
- Performance

Byzantine general problem:
Distributing a computation over several machines (worker processes) provides a higher risk of failures. These include crashes and computation errors, stalled processes, biases in the way the data samples are distributed among the processes, but also, in the worst case, attackers trying to compromise the entire system. The most robust system is one that tolerates Byzantine failures i.e., completely arbitrary behaviors of some of the processes. A classical approach to mask failures in distributed systems is to use a state machine replication protocol , which requires however state transitions to be applied by all worker processes. In the case of distributed machine learning, this constraint can be translated in two ways: either (a) the processes agree on a sample of data based on which they update their local parameter vectors, or (b) they agree on how the parameter vector should be updated.

Heterogeneity:
The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks. Heterogeneity applies to all of the following:

- Hardware devices: computers, tablets, mobile phones, embedded devices, etc.
- Operating System: MS Windows, Linux, Mac, Unix, etc.
- Network: Local network, the Internet, wireless network, satellite links, etc.
- Programming languages: Java, C/C++, Python, PHP, etc.
- Different roles of software developers, designers, system managers

Heterogeneity comes from several factors:

- Users, applications and data
- Software and the hardware on which it runs
- Interconnection networks
- Organizations

Transparency:
T transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole rather than as a collection of independent components. In other words, distributed systems designers must hide the complexity of the systems as much as they can

Concurrency:
Both services and applications provide resources that can be shared by clients in a distributed system. There is therefore a possibility that several clients will attempt to access a shared resource at the same time. For example, a data structure that records bids for an auction may be accessed very frequently when it gets close to the deadline time. For an object to be safe in a concurrent environment, its operations must be synchronized in such a way that its data remains consistent. This can be achieved by standard techniques such as semaphores, which are used in most operating systems.

Security: Many of the information resources that are made available and maintained in distributed systems have a high intrinsic value to their users. Their security is therefore of considerable importance. Security for information resources has three components:

- Confidentiality (protection against disclosure to unauthorized individuals)
- Integrity (protection against alteration or corruption),
- Availability for the authorized (protection against interference with the means to access the resources).

Scalability:
Distributed systems must be scalable as the number of user increases. The scalability is defined by B. Clifford Neuman as a system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity

Failure Handling:
Computer systems sometimes fail. When faults occur in hardware or software, programs may produce incorrect results or may stop before they have completed the intended computation. The handling of failures is particularly difficult.

Quality of service:
Once users are provided with the functionality that they require of a service, such as the file service in a distributed system, we can go on to ask about the quality of the service provided. The main nonfunctional properties of systems that affect the quality of the service experienced by clients and users are reliability, security and performance. Adaptability to meet changing system configurations and resource availability has been recognized as a further important aspect of service quality.

Reliability:
One of the original goals of building distributed systems was to make them more reliable than single-processor systems. The idea is that if a machine goes down, some other machine takes over the job. A highly reliable system must be highly available, but that is not enough. Data entrusted to the system must not be lost or garbled in any way, and if files are stored redundantly on multiple servers, all the copies must be kept consistent. In general, the more copies that are kept, the better the availability, but the greater the chance that they were be inconsistent, especially if updates are frequent.

Performance:
Always the hidden data in the background is the issue of performance. Building a transparent, flexible and reliable distributed system the more important is in its performance. In particular, when running a particular application on a distributed system, it should not be appreciably worse than running the same application on a single processor. Unfortunately, achieving this is easier said than done.

III CONCLUSIONS

Distributed systems are very challenging field in research based environment and in industrial environment. They are ly complex to develop and maintain and can be error prone. As the amount of data grows, efficiently and effectively mining information from data becomes very challenging. Fundamental issues arise when figuring out how to make the computations, storage, transmission, and imputation of data happen in a low-latency way. Distributed systems could easily be justified by the simple facts of collaboration and sharing. The world-wide web is an obvious and compelling example of the value that is created when people can easily expose and exchange information. Much of the work we do is done in collaboration with others, and so we need often need to share work products.

IV REFERENCES

[1] Bekkerman, R., Bilenko, M., and Langford, J. 2011 Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.

[2] Ben-Tal, A. and Nemirovski A. 2001 Lectures on modern convex optimization: analysis, algorithms, and engineering applications, volume 2. Siam.

[3] Bernstein, J. , XiangWang, Y., Azizzadenesheli, K. and Anandkumar. A. 2018 Signsgd: Compressed optimi sation for non-**convex** problems. In International Conference on Machine Learning, pages 559–568.

[4] Bijral, A. S., Sarwate, Anand D., and Srebro N. 2016, On data dependence in distributed stochastic optimization. arXiv preprint arXiv:1603.04379..

[5] 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, pages 1531–1539.

[6] Hazan., E.. Introduction to online convex optimization. Foundations and Trends® in Optimization, 2016 , 2(3-4): 157–325.

[7] 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, pages 770–778,

[8] 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.

[9] Shwartz, S. S. and David, S. B. , 2014, Understanding machine learning: From theory to algorithms. Cambridge university pres.

[10] Zinkevich, M., 2003, Online convex programming and generalized infinitesimal gradient ascent. In International Conference on Machine Learning, pages 928–936.

[11] HATCHER, W. G., AND YUA, W., 2018, Survey of Deep Learning: Platforms,Applications and Emerging Research Trends, IEEE Access, May 24.

[12] Chen X.W. and Lin X., 2016, ``Big data deep learning: Challenges and perspectives," IEEE Access, vol. 2, pp. 514525, 2014. 14. Y. Ding, S. Chen, and J. Xu, ``Application of deep belief networks for opcode based malware detection," in Proc. Int. Joint Conf. Neural Netw. (IJCNN), pp. 39013908.

[13] 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), pp. 310313.

[14] Zhu, D., Jin, H., Y, Y., Wu, D. and Chen, W., 2017, ``DeepFlow: Deep learningbased malware detection by mining Android application for abnormal usage of sensitive data," in Proc. IEEE Symp. Comput. Commun. (ISCC), pp. 438443.

[15] Ujjwalkarn, 2016, An Intuitive Explanation of Convolutional Neural Networks. [online].Available: https://ujjwalkarn.me/2016/08/11/intuitiveexplanation- convnets/

[16] Nielsen, M., 2018. Neural Networks and Deep Learning. [Online]. Available: 19. Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In Advances in Neural Information Processing Systems,

pages 4613–4623.

[17] Blanchard, P., Guerraoui, R., and Stainer, J. , 2017, Machine learning with adversaries: Byzantine tolerant gradient descent. In Advances in Neural Information Processing Systems, pages 119–129.

[18] 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 roceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, OSDI'14, pages 583–598, Berkeley, CA, USA, 2014. USENIX Association.

[19] McMahan B. and Ramage, R., 2017, Federated learning: Collaborative machine learning without centralized training data. Google Research Blog, 3.

[20] Yin, D., Chen, Y., Ramchandran, R. and Bartlett. Byzantine B., 2018, robust distributed learning: Towards optimal statistical rates. In Proceedings of the International Conference on Machine Learning (ICML), pages 5636–5645.