Analysis of Distributed Systems

Ehsan Bazgir
School of Engineering
San Francisco Bay University
Fremont, CA 94539, USA

Tasmita Tanjim Tanha
Department of Computer Science, School of Engineering,
San Francisco Bay University,
Fremont, CA, 94539, USA

Anwarul Azim Bhuiyan
Dept. of EEE
East West University,
Dhaka 1212, Bangladesh

Ehteshamul Haque
Department of Computer Science,
School of Engineering,
San Francisco Bay University, Fremont, CA 94539, USA.

Md Shihab Uddin
School of Engineering
San Francisco Bay University
Fremont, CA 94539, USA

ABSTRACT
Distributed computing systems have become a pivotal aspect of modern technology, enabling complex computations and tasks across various industries, including telecommunications, scientific research, and financial services. These systems provide scalability, fault tolerance, and resource sharing across multiple nodes, allowing for efficient handling of vast datasets. This paper discusses the characteristics, architectural styles, and middleware solutions that support distributed computing, with a focus on fault tolerance mechanisms and distributed consensus algorithms such as Paxos and Raft. Additionally, it compares notable frameworks like Apache Spark and Ray, which serve distinct roles in managing data processing and real-time computational tasks, respectively. The discussion highlights the importance of fault tolerance and resilience in ensuring the continuous operation of distributed systems in diverse applications.

Keywords: Distributed Systems, Fault Tolerance, Scalability, MapReduce, Apache Spark, Ray Framework. Middleware, Distributed File Systems, Paxos Algorithm

1. INTRODUCTION
Distributed computing is used now in a wide range of computing activities, from video games to database management. Without distributed computing, many software applications—such as blockchain technology, scientific simulations, cryptocurrency systems, and AI platforms—would just not be possible to implement [1].

When a burden is too great for one computer or device to manage, distributed systems kick in. When workloads fluctuate, such on Cyber Monday when e-commerce traffic surges or when news about your company causes an increase in online traffic, they are essential [21].

Using the capabilities of several computers and processes, distributed systems can offer features that would be difficult or impossible to do on a single system. They enable, for example, off-site server and application backups, which allow the main catalog to ask the off-site node or nodes to deliver the segments in case it is short on bits for a restoration. Whether sending an email, playing a game, or reading this article online, almost every activity done on a computing device uses the features of distributed systems.

Many different industries have a wide range of distributed system examples, such as: - Telecommunications networks supporting internet and mobile networks Systems of graphic and video rendering - Scientific computation in fields like genetics and protein folding System of hotel and airline reservations Systems of multiuser video conferences Systems for processing cryptocurrency, such as Bitcoin; peer-to-peer file-sharing Multiple player video games - Distributed community compute systems Distribution of merchants around the world and supply chain management.

2. CHARACTERISTICS OF DISTRIBUTED COMPUTING SYSTEMS

2.1 Collection of autonomous computing elements
Today's distributed systems may consist of various nodes, ranging from powerful computers to small devices. The primary concept is that these nodes can function independently, even though collaboration is necessary for the system to operate efficiently. Nodes are programmed to work towards shared objectives by exchanging messages. Each node responds to incoming messages, processes them, and initiates further communication using message passing.

Due to the autonomy of nodes, each one has its own concept of time, resulting in a lack of a universal clock. This time disparity raises important issues related to synchronization and coordination in distributed systems. Managing a group of nodes involves handling membership and organization. This includes determining which nodes are part of the system and providing each member with a list of nodes for direct communication.

2.2 Single coherent system
A distributed system should present itself as a unified and coherent entity, even though processes, data, and control are spread across a network. While achieving a seamless single-system view may be too ambitious, our definition focuses on the system appearing coherent. Essentially, a distributed system is considered coherent if it functions as users expect. In a single coherent system, all nodes collectively operate consistently, regardless of the location, time, or method of user interaction.

Providing a unified view is often a difficult task. It involves ensuring that users cannot discern where a process is running, or if parts of a task have been delegated to other processes elsewhere. Users should not need to know where data is stored or be concerned with data replication for performance improvement. This concept of distribution transparency is a key objective in distributed system design, reminiscent of how Unix-like operating systems offer a unified file-system interface to abstract differences between various resources.

2.3 Middleware and distributed systems

Middleware in distributed systems plays a crucial role in assisting the development of distributed applications by serving as a separate layer of software placed on top of the respective operating systems of the computers within the system. This layer, known as middleware, acts as a manager of resources that allows applications to efficiently share and deploy resources across the network. In addition to resource management, middleware provides services such as inter application communication, security, accounting, and failure recovery. Unlike operating systems, middleware operates in a networked environment. YARN is a popular middleware [3], a framework for resource management and task scheduling.

2.4 Fault Tolerant

Fault tolerance is a key feature of distributed systems, ensuring that they remain functional even in the event of node failures. This resilience is achieved through redundancy, where critical data or services are duplicated across multiple nodes. Redundant components can seamlessly take over tasks from failed nodes, minimizing downtime and preventing data loss. Fault tolerance is particularly important for applications that require high availability, such as financial transaction systems, healthcare data services, and critical infrastructure monitoring.

3. Architectural Styles in Distributed Systems

Various architectural styles are utilized in distributed systems, with each one being customized to address specific needs and obstacles. This discourse delves into three notable architectures, such as MapReduce, Apache Spark, and the Google File System (GFS), showcasing their respective characteristics. These architectures are all geared towards effectively handling and overseeing extensive data sets spread across computer clusters [4].

3.1 MapReduce

The distributed file system divides large data files, while the MapReduce programming model splits the algorithm into segments that can be processed on data blocks in a distributed manner to achieve optimal computing performance. Initially created by Google, MapReduce was later integrated into the Apache Hadoop project to transform sequential algorithms into a MapReduce format, enabling efficient execution on a cluster [5, 6].

A MapReduce program consists of two fundamental processes: Map and Reduce [7]. The Map process operates on the data blocks of a node to produce a local outcome. This process is carried out concurrently on multiple nodes to generate local results independently, thereby enhancing computing performance [8]. The Reduce process acts on the local outcomes to produce a global result. This stage involves transferring all local results to the nodes responsible for the Reduce process, incurring a high data communication cost due to shuffling and transforming data among nodes. Upon gathering all local results at the Reduce nodes, a global result is produced through the Reduce process [9].

When a data processing task can be accomplished with a single set of Map and Reduce operations, such as tallying word frequencies from numerous web pages, the MapReduce program can effectively analyze large data files by leveraging a large-scale cluster with numerous nodes. Conversely, if an iterative algorithm is converted into a series of Map and Reduce operations, the algorithm may not efficiently handle a vast distributed dataset due to factors like I/O, communication, and computing expenses [10].

3.2 Apache Spark

Apache Spark, originally created at the University of California, Berkeley, is an open-source engine for processing large-scale data. It differs from Hadoop MapReduce by storing all interim results in a Resilient Distributed Dataset (RDD) in memory to reduce I/O costs. It also employs a directed acyclic graph (DAG) task segmentation method for operating on RDD, similar to MapReduce. Spark's in-memory computing outperforms Hadoop, making it the leading platform for batch big data analysis [11-13].

3.3 Distributed File Systems

By employing the divide-and-conquer approach in distributed computing, a significant data file gets divided into several small files known as data blocks. These data blocks are then distributed across cluster nodes' disks to enhance I/O performance. This method of storing a large data file is termed as a distributed data file and is effectively managed on the cluster using various distributed file systems [14, 15] like GFS [16], HDFS [17], TFS [18], and FastDFS [19]. These file systems play a crucial role in facilitating big data analysis. Figure 1 describes distributed Computing Frameworks for big data analysis.

GFS, a Linux-based distributed file system developed by Google, caters to the specific needs of individual companies [20]. TFS, on the other hand, is a high-availability, high-performance distributed file system created by Taobao to address the storage demands of unstructured small files, usually under 1 MB in size. FastDFS, an open-source distributed file system, is a lightweight option ideal for online services that utilize files as their primary medium.

HDFS, originating from the Apache Hadoop project, was crafted to tackle the complexities of distributed data processing within extensive clusters. It serves as a fault-tolerant data storage system running on standard hardware, making it well-suited for managing large volumes of big data. As a result, HDFS has gained wide acceptance in the industry for its role in processing and analyzing big data.

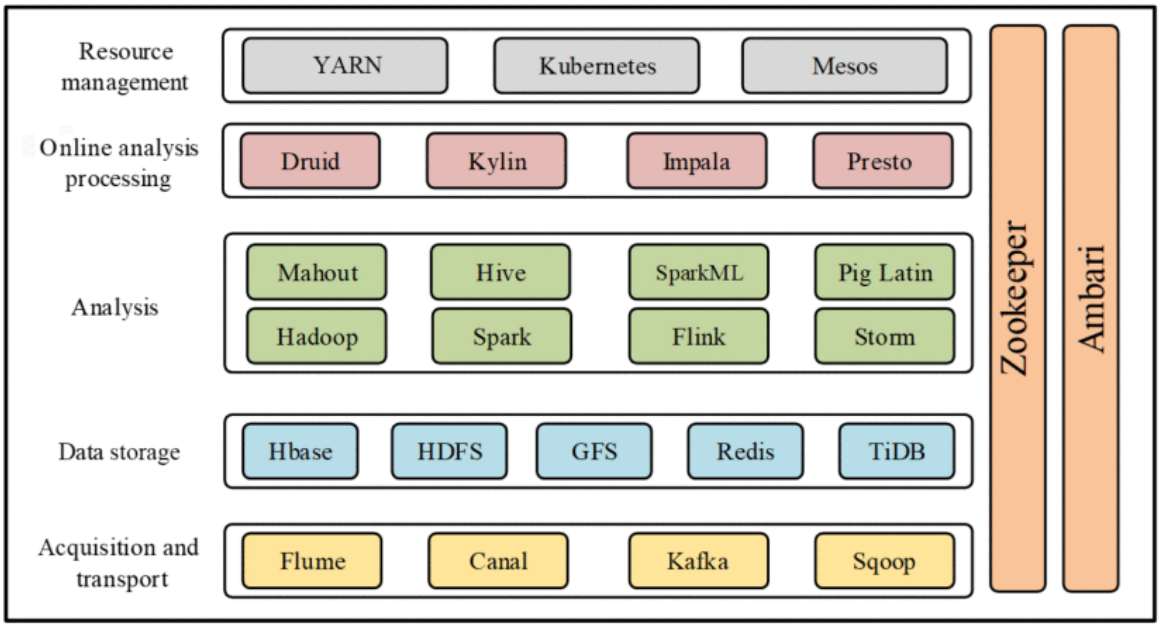

Fig. 1: Distributed Computing Frameworks for Big Data Analysis [4]

4. FAULT TOLERANCE AND RESILIENCE

Distributed architecture has the major benefit of fault tolerance. Continuous service is ensured by the ability of other nodes to take over for a failing one. Large, complicated applications find this architecture to be a great option because it can be readily scaled up by adding more machines.

4.1 Fault Tolerance Mechanisms

Fault tolerance refers to the capacity of a system to function normally even if some of its components fail. Fault tolerance in distributed systems is frequently accomplished by use of redundancy, replication, and error recovery techniques.

Replication: Replication is keeping copies of the same data or service on several nodes so that the system may still access the data from another node in the event that one fails. File and database systems both availability [22].

Redundancy: Hardware, network, and data layers of a distributed system can all have redundancy implemented. It is building up backup copies of system components, including servers or hard drives, that can take over when the main one malfunctions [22].

Error Recovery: Error recovery methods are intended to put the system back to a known good state after a failure. This might entail processes like checkpointing, in which the system routinely stores a state snapshot so that, in the event of a crash, it can resume from this point instead of having to start over [22].

4.2 Techniques for Detecting and Handling Failures

Detecting and handling failures are critical for maintaining system reliability and availability in distributed environments.

(a) Heart beating: A heartbeat is a periodic signal sent between machines to indicate normal operation. If a heartbeat is missed, other components in the system can assume that a failure has occurred. This method is simple and effective for monitoring the health of nodes in distributed systems [23].

(b) Timeouts: Timeouts are used to detect failures by setting a time limit on certain operations or responses. If the expected response is not received within the timeout period, the system can assume a failure and initiate appropriate recovery or failover procedures [23].

(c) Failure Detectors: Failure detectors help in identifying component failures within a system. They can vary in accuracy and speed, and are crucial for deciding when to trigger a failover or recovery process [23].

4.3 Algorithms for Distributed Consensus

In distributed systems, consensus methods are necessary to guarantee that every node agrees on a single data value or a single series of events, which is critical to preserving consistency among dispersed operations.

Fig. 2: Consensus Algorithm in Distributed System [24]

4.3.1 Paxos: It allows a cluster of distributed database nodes or other dispersed group of computers to come to an agreement via an asynchronous network. One or more of the computers offers Paxos a value in order to reach an agreement. When most of the Paxos-running computers concur on a given value, consensus is reached. Paxos chooses a single value from among one or more suggested values and broadcasts it to every cooperating computer. The cluster clocks ahead once every computer (or database node) agrees on the suggested value after the Paxos process has completed [24].

4.3.2 Raft: Raft has been designed to be more comprehensible than Paxos and functions on the basis of a robust leader concept. The system is partitioned into three primary constituents: Leader Election, Log Replication, and Safety.

5. RAY FRAMEWORK

Ray is a framework that is open-source and offers a straightforward, universal API for constructing distributed applications. Ray is specifically engineered to deliver exceptional performance and scalability, especially for applications that require advanced computational capabilities, such as machine learning and artificial intelligence [29]. Figure 6 and 7 describes Ray architecture.

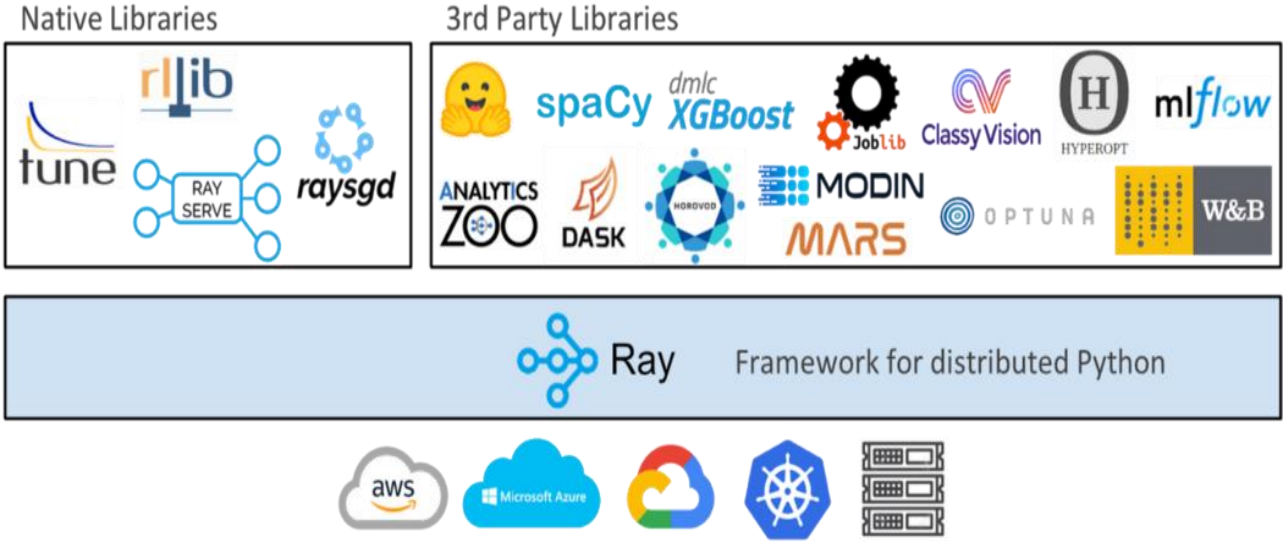

Fig. 6: Ray ecosystem [30]

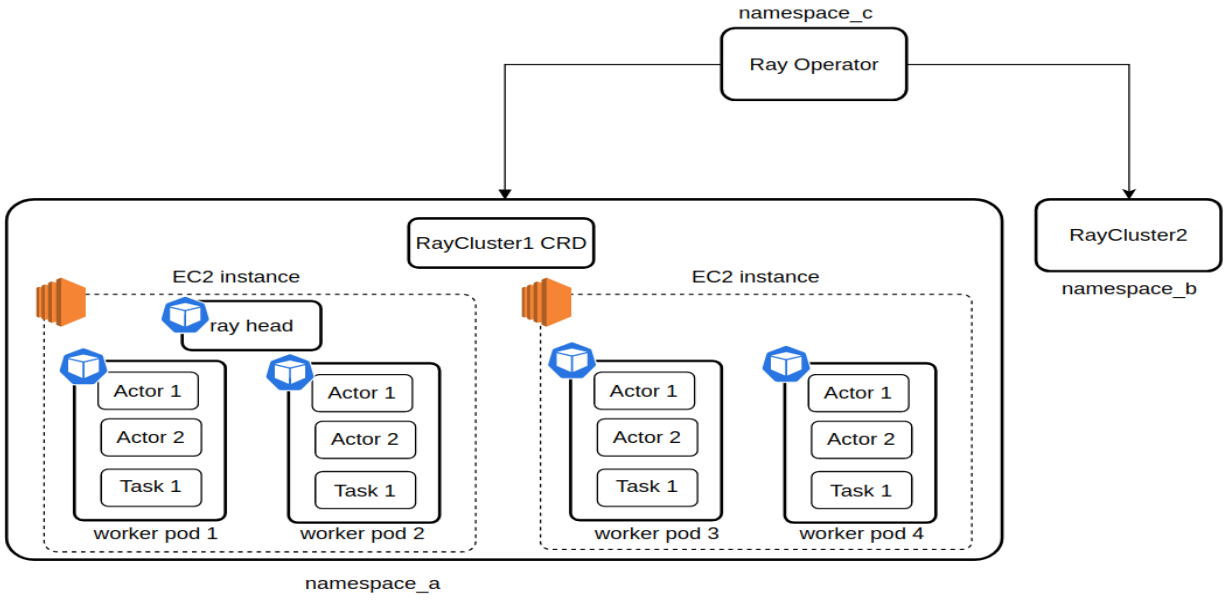

Fig. 7: Ray Core [30]

5.1 PERFORMANCE CHARACTERISTICS

(a) High Throughput and Low Latency: Ray's task execution framework is optimized to provide high throughput and low latency in task scheduling and execution. This makes it well-suited for applications that require high performance.

(b) Scalability: Ray has the potential to easily handle hundreds to thousands of nodes in a horizontal manner, enabling applications to efficiently utilize more computational resources as required, without seeing a substantial decline in performance.

5.2 Comparison between Spark and Ray

Spark is primarily built for data processing workflows and batch processing. It also supports streaming data through micro-batching. It is very suitable for tasks such as ETL workloads, batch queries, and data transformation workflows.

Ray is specifically designed for applications that require real-time processing and high speed. It has built-in support for both batch processing and streaming data. It performs exceptionally well in situations that need immediate decision-making and interactive computing.

Spark utilizes a resilient distributed dataset (RDD) and directed acyclic graph (DAG) to execute tasks. However, this approach may not be as effective for iterative algorithms that involve managing a large amount of mutable state.

Ray supports the execution of dynamic task graphs, which can be more efficient for applications that require frequent modifications to state or that benefit from precise task management.

Spark executes computations in memory, and its efficiency greatly depends on memory management and the capability to store datasets in memory throughout the cluster.

Ray utilizes an object store to manage shared memory among activities, hence minimizing the costs associated with data transportation and duplication.

Spark and Ray can be used synergistically in IoT applications. Spark can handle the initial stages of data ingestion, cleaning, and aggregation, while Ray can focus on real-time processing, decision-making, and AI model deployment. By leveraging the strengths of both frameworks, IoT systems can achieve scalable data handling, robust analytics, and dynamic performance optimization, addressing the diverse demands of modern IoT ecosystems [31-36].

Distributed systems can integrate AI to provide predictive analytics, automated diagnosis, and optimized treatment plans. Blockchain offers secure, decentralized data management and ensures the integrity of healthcare records. 5G enhances the connectivity and reliability of IoT devices in distributed systems, enabling real-time data transmission [37-45].

6. CONCLUSION

Distributed systems play an essential role in enabling scalable and fault-tolerant solutions for handling large datasets and complex computations. Through mechanisms like replication, redundancy, and failure detection, these systems ensure high availability and resilience. The architectural styles explored, including MapReduce, Spark, and the Google File System, provide a strong foundation for big data processing. As technology continues to evolve, frameworks such as Ray offer enhanced capabilities for real-time processing and machine learning applications, making distributed computing indispensable in the modern digital landscape. Future advancements will likely focus on improving fault tolerance, reducing latency, and enhancing scalability to meet the growing demands of data-intensive industries.

7. REFERENCES

[1] https://www.splunk.com/en_us/blog/learn/distributed-systems.html#:~:text=Distributed%20systems%20are%20used%20when,to%20news%20about%20your%20organization.

[2] van Steen, M., Tanenbaum, A.S. A brief introduction to distributed systems. Computing 98, 967–1009 (2016). https://doi.org/10.1007/s00607-016-0508-7

[3] P. S. Janardhanan and P. Samuel, "Launch overheads of spark applications on standalone and hadoop YARN clusters" in Advances in Electrical and Computer Technologies, Singapore:Springer, pp. 47-54, 2020.

[4] X. Sun, Y. He, D. Wu and J. Z. Huang, "Survey of Distributed Computing Frameworks for Supporting Big Data Analysis," in Big Data Mining and Analytics, vol. 6, no. 2, pp. 154-169, June 2023, doi: 10.26599/BDMA.2022.9020014.

[5] R. Gu, X. Yang, J. Yan, Y. Sun, B. Wang, C. Yuan, et al., "SHadoop: Improving MapReduce performance by optimizing job execution mechanism in hadoop clusters", J. Parallel Distribut. Comput., vol. 74, no. 3, pp. 2166-2179, 2014.

[6] I. Polato, R. Ré, A. Goldman and F. Kon, "A comprehensive view of hadoop research-A systematic literature review", J. Network Comput. Applicat., vol. 46, pp. 1-25, 2014.

[7] Y. Wang, W. Jiang and G. Agrawal, "SciMATE: A novel MapReduce-like framework for multiple scientific data formats", Proc. 2012 12 th IEEE/ACM Int. Symp. Cluster Cloud and Grid Computing (CCGRID 2012) , pp. 443-450, 2012.

[8] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters", Commun ACM, vol. 51, no. 1, pp. 107-113, 2008.

[9] M. R. Ghazi and D. Gangodkar, "Hadoop MapReduce and HDFS: A developers perspective", Proc. Comput. Sci., vol. 48, pp. 45-50, 2015.

[10] Y. Zhang, Q. Gao, L. Gao and C. Wang, "iMapReduce: A distributed computing framework for iterative computation", J. Grid Comput., vol. 10, no. 1, pp. 47-68, 2012.

[11] J. Yu, J. Wu and M. Sarwat, "A demonstration of geoSpark: A cluster computing framework for processing big spatial data", Proc. 2016 IEEE 32 nd Int. Conf. Data Engineering (ICDE) , pp. 1410-1413, 2016.

[12] Z. Yang, C. Zhang, M. Hu and F. Lin, "OPC: A distributed computing and memory computing-based effective solution of big data", Proc. 2015 IEEE Int. Conf. Smart City/ SocialCom/SustainCom (SmartCity), pp. 50-53, 2015.

[13] V. Taran, O. Alienin, S. Stirenko, Y. Gordienko and A. Rojbi, "Performance evaluation of distributed computing environments with Hadoop and spark frameworks", Proc. 2017 IEEE Int. Young Scientists Forum on Applied Physics and Engineering (YSF), pp. 80-83, 2017.

[14] T. D. Thanh, S. Mohan, E. Choi, S. Kim and P. Kim, "A taxonomy and survey on distributed file systems", Proc. 2008 4 th Int. Conf. Networked Computing and Advanced Information Management , pp. 144-149, 2008.

[15] J. Blomer, "A survey on distributed file system technology", J. Phys. Conf. Ser., vol. 608, pp. 012039, 2015.

[16] S. Ghemawat, H. Gobioff and S. T. Leung, "The google file system", ACM SIGOPS Oper. Syst. Rev., vol. 73, no. 5, pp. 29-43, 2003.

[17] L. Jiang, B. Li and M. Song, "The optimization of HDFS based on small files", Proc. 2010 3 rd IEEE Int. Conf.

Broadband Network and Multimedia Technology (IC-BNMT) , pp. 912-915, 2010.

[18] S. Zhuo, X. Wu, W. Zhang and W. Dou, "Distributed file system and classification for small images", Proc. 2013 IEEE Int. Conf. Green Computing and Communications and IEEE Internet of Things and IEEE Cyber Physical and Social Computing, pp. 2231-2234, 2013.

[19] H. Che and H. Zhang, "Exploiting fastDFS client-based small file merging", Proc. 2016 Int. Conf Artificial Intelligence and Engineering Applications, pp. 242-246, 2016.

[20] Z. Ullah, S. Jabbar, M. H. Bin, Tariq Alvi and A. Ahmad, "Analytical study on performance challenges and future considerations of Google file system", Int. J. Computer Communicat. Eng., vol. 3, no. 4, pp. 279-284, 2014.

[21] https://medium.com/@ayeshwery/architectures-in-distributed-system-b2ace2fca6bb

[22] Tanenbaum, A.S., & Van Steen, M. (2017). "Distributed Systems: Principles and Paradigms."

[23] Chandra, T.D., & Toueg, S. (1996). "Unreliable Failure Detectors for Reliable Distributed Systems."

[24] https://medium.com/@mani.saksham12/raft-and-paxos-consensus-algorithms-for-distributed-systems-138cd7c2d35a

[25] Ongaro, D., & Ousterhout, J. (2014). "In Search of an Understandable Consensus Algorithm."

[26] https://kafka.apache.org/documentation/

[27] https://medium.com/@kajol_singh/unveiling-apache-kafka-a-comprehensive-guide-to-core-concepts-and-functionality-2efd51de2b89

[28] https://bair.berkeley.edu/blog/2018/01/09/ray/

[29] Moritz, Philipp, et al. "Ray: A distributed framework for emerging {AI} applications." 13th USENIX symposium on operating systems design and implementation (OSDI 18). 2018.

[30] https://www.datacamp.com/tutorial/distributed-processing-using-ray-framework-in-python

[31] Hoque, K., Hossain, M. B., Sami, A., Das, D., Kadir, A., & Rahman, M. A. (2024). Technological trends in 5G networks for IoT-enabled smart healthcare: A review. International Journal of Science and Research Archive, 12(2), 1399-1410.

[32] Md Shihab Uddin. Addressing IoT Security Challenges through AI Solutions. International Journal of Computer Applications. 186, 45 (Oct 2024), 50-55. DOI=10.5120/ijca2024924107

[33] Khandoker Hoque, Md Boktiar Hossain, Denesh Das, Partha Protim Roy. Integration of IoT in Energy Sector. International Journal of Computer Applications. 186, 36 (Aug 2024), 32-40. DOI=10.5120/ijca2024923981

[34] Md Maniruzzaman, Md Shihab Uddin, Md Boktiar Hossain, Khandoker Hoque, "Understanding COVID-19 Through Tweets using Machine Learning: A Visualization of Trends and Conversations", European Journal of Advances in Engineering and Technology, Vol. 10, Issue: 5, pp. 108-114, 2023.

[35] Md Boktiar Hossain, Khandoker Hoque, Mohammad Atikur Rahman, Priya Podder, Deepak Gupta, "Hepatitis C Prediction Applying Different ML Classification Algorithm", International Conference on Computing and Communication Networks 2024 (ICCCNet 2024), 2024.

[36] Javed Mehedi Shamrat, F. M., Tasnim, Z., Chowdhury, T. R., Shema, R., Uddin, M. S., & Sultana, Z. (2022). Multiple cascading algorithms to evaluate performance of face detection. In Pervasive Computing and Social Networking: Proceedings of ICPCSN 2021 (pp. 89-102). Springer Singapore.

[37] Javed Mehedi Shamrat, F. M., Ghosh, P., Tasnim, Z., Khan, A. A., Uddin, M. S., & Chowdhury, T. R. (2022). Human Face recognition using eigenface, SURF method. In Pervasive Computing and Social Networking: Proceedings of ICPCSN 2021 (pp. 73-88). Springer Singapore.

[38] Kowsher, M., Tahabilder, A., Sanjid, M. Z. I., Prottasha, N. J., Uddin, M. S., Hossain, M. A., & Jilani, M. A. K. (2021). LSTM-ANN & BiLSTM-ANN: Hybrid deep learning models for enhanced classification accuracy. Procedia Computer Science, 193, 131-140.

[39] Mondai, R., & Rahman, M. M. (2017, July). Dynamic analysis of variable structure based sliding mode intelligent load frequency control of interconnected nonlinear conventional and renewable power system. In 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT) (pp. 393-400). IEEE.

[40] Bharati, S., Rahman, M. A., Mondal, R., Podder, P., Alvi, A. A., & Mahmood, A. (2020). Prediction of energy consumed by home appliances with the visualization of plot analysis applying different classification algorithm. In Frontiers in Intelligent Computing: Theory and Applications: Proceedings of the 7th International Conference on FICTA (2018), Volume 2 (pp. 246-257). Springer Singapore.

[41] Hoque, R., Maniruzzaman, M., Michael, D. L., & Hoque, M. (2024). Empowering blockchain with SmartNIC: Enhancing performance, security, and scalability. World Journal of Advanced Research and Reviews, 22(1), 151-162.

[42] Amit Deb Nath, Rahmanul Hoque, Md. Masum Billah, Numair Bin Sharif, Mahmudul Hoque. Distributed Parallel and Cloud Computing: A Review. International Journal of Computer Applications. 186, 16 (Apr 2024), 25-32. DOI=10.5120/ijca2024923547

[43] Maniruzzaman, M., Sami, A., Hoque, R., & Mandal, P. (2024). Pneumonia prediction using deep learning in chest X-ray Images. International Journal of Science and Research Archive, 12(1), 767-773.

[44] M. S. Miah and M. S. Islam, "Big Data Analytics Architectural Data Cut off Tactics for Cyber Security and Its Implication in Digital forensic," 2022 International Conference on Futuristic Technologies (INCOFT), Belgaum, India, 2022, pp. 1-6, doi: 10.1109/INCOFT55651.2022.10094342.

[45] Obaida, M. A., Miah, M. S., & Horaira, M. A. (2011). Random Early Discard (RED-AQM) Performance Analysis in Terms of TCP Variants and Network Parameters: Instability in High-Bandwidth-Delay Network. International Journal of Computer Applications, 27(8), 40-44.