International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Exploring Hadoop: Distributed Storage to Advanced Analytics

Sanskruti .A. Waghmare
Department of Computer Science and Engineering
Sipna College of Engineering and Technology
Amravati, Maharashtra, India 444603

***Abstract*—** With the growing number of connected devices, sensors and enterprise systems, we are generating unprecedented amounts of data that are referred to as "big data." To meet this growth, Apache Hadoop was born, and it offers distributed storage and parallel processing over clusters of commodity hardware. While Hadoop is well established, most of the existing literature focuses on individual components or applications, not the whole framework and few give credit to the Apache Nutch project, and do not explain why it's called Hadoop. This review first covers the historical and conceptual evolution of Hadoop, and then systematically reviews the fundamental components of Hadoop: the Hadoop Distributed File System (HDFS), MapReduce and YARN, and the major ecosystem tools: Hive, Pig, HBase, Sqoop, ZooKeeper, Mahout, Spark and Flume. The paper also explores the security aspects of Hadoop, how it's being used in the industry, and how it has become a platform to power advanced analytics such as data mining, machine learning, NLP, vectorization, predictive analytics, etc. The review ends with a set of open research questions and future directions, providing a single point of contact for novice and new researchers looking to further develop Hadoop to support more sophisticated analytics.

Index Terms*—Apache Hadoop, HDFS, MapReduce, YARN, distributed computing, big data ecosystem, Hadoop security, advanced analytics.*

I. INT DUCTION

The growth of information and communication technologies, sensing systems and knowledge-discovery methods has fuelled a deluge of data generated by smart devices, enterprise applications and cloud platforms. As organizations increasingly rely on this data for diagnostic, predictive and prescriptive insight, scalable and cost-effective processing frameworks have become essential. Apache Hadoop is among the most significant responses to this demand [13].

Hadoop is an open-source framework for storing and processing large datasets across distributed, commodity-hardware clusters, built around the MapReduce programming model. Rather than relying on expensive high-end machines, Hadoop achieves scalability, fault tolerance and cost efficiency by clustering many inexpensive nodes. It originated from work by Doug Cutting and Mike Cafarella at Yahoo on the Nutch search engine; the storage-and-processing component was later named "Hadoop" after Cutting's son's toy elephant.

Hadoop is a registered trademark of the Apache Software Foundation (ASF), which governs it and the surrounding ecosystem—Hive, Pig, HBase, Spark, Sqoop, Oozie and ZooKeeper among them—under a meritocratic, community-driven model [13]. Every ASF project is released under the Apache License 2.0, permitting free use, modification and commercial distribution, which keeps the ecosystem vendor-neutral and lets independently developed tools interoperate through shared metadata stores and resource-management layers. Because Hadoop began as the work of only two engineers, its placement under the ASF ensured that the community could sustain and extend it indefinitely, which is why the modern Hadoop ecosystem is best understood as a family of Apache-branded projects sharing common governance, licensing and quality standards.

Traditional relational database systems were designed for structured, moderate-volume data and struggle with the volume, velocity and variety that characterize modern big data workloads—vertical scaling on a single high-end server quickly becomes cost-prohibitive and eventually hits a physical ceiling. Instead, Hadoop scales horizontally—by adding nodes, storage and processing capacity grows roughly linearly, and the data is replicated across nodes by HDFS, and failed tasks are automatically rescheduled by MapReduce/YARN. This combination of horizontal scalability, fault tolerance and commodity-hardware economics is what allowed Hadoop to displace single-server and specialized-hardware approaches for large-scale batch analytics, and it remains the foundation on which the rest of the ecosystem—and the advanced-analytics capabilities discussed later in this paper—is built.

The remainder of this paper is organized as follows. Section II traces Hadoop's history from the Nutch project to Hadoop 3.0. Section III details the core architecture—HDFS, MapReduce, YARN and Hadoop Common. Section IV surveys the principal ecosystem tools. Section V examines Hadoop's security landscape. Section VI reviews industry applications, and Section VII surveys Hadoop's role in advanced analytics. Section VIII discusses the research gap this review addresses together with open challenges and future directions, Section IX synthesizes these findings in a discussion, and Section X concludes the paper.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

II. HISTORY OF HADOOP

The Nutch project, an open-source web search engine started in 2002 by Cutting and Cafarella, needed to crawl and index billions of pages—a scale that demanded distributed storage and parallel processing beyond the technology of the time. Google's 2003 paper on the Google File System (GFS) and its 2004 MapReduce paper directly inspired the Nutch developers to build their own distributed file system and processing model, which evolved into Hadoop [13].

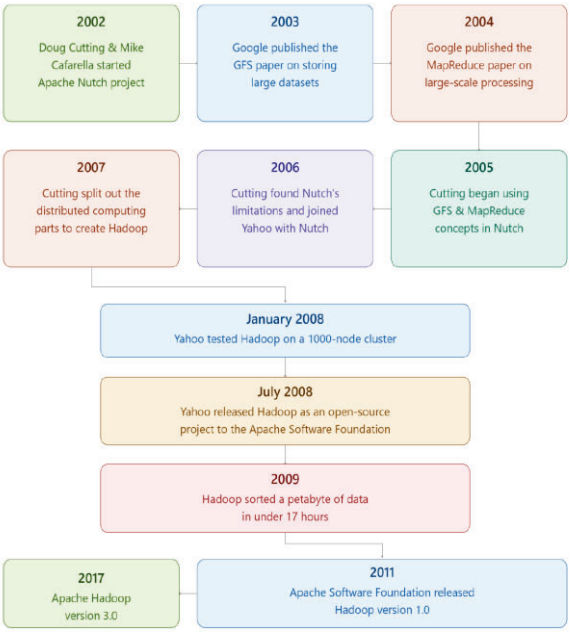

Fig. 1. Hadoop development timeline, from the Apache Nutch project (2002) to Hadoop 3.0 (2017).

Nutch split from Hadoop in 2006 as Yahoo took over its development and scaled it to hundreds of nodes. In 2007 Yahoo deployed Hadoop on a 1,000-node cluster, demonstrating its scalability and fault tolerance, and in 2008 Hadoop became a top-level ASF project. By 2009 it was benchmarked sorting one petabyte of data in 17 hours, establishing it as an enterprise-scale solution. Apache Hadoop 1.0 (2011) stabilized HDFS and MapReduce, marking the framework's mature, widely-adopted phase. Hadoop 2 introduced YARN as a separate resource-management layer, decoupling scheduling from MapReduce and enabling multiple processing frameworks to share a cluster. Hadoop 3.0 (2017) added multiple NameNodes for high availability, erasure coding for efficient storage, enhanced YARN resource management, and integration with cloud and containerized environments [15]. Table I summarizes the resulting evolution across the three major release lines.

TABLE I
EVOLUTION OF HADOOP MAJOR RELEASES

Version	Core Layers	Key Change
1.0 (2011)	HDFS + MapReduce	Single NameNode (single point of failure); JobTracker/TaskTracker scheduling
2.0 (2013)	+ YAR	Resource management separated from job scheduling; multiple processing frameworks per cluster
3.0 (2017)	+ HA erasure coding	Multiple NameNodes; erasure coding for storage efficiency; container/cloud integration; GPU support

III. ADOOP ARCHITECTURE

A. Hadoop Distributed File System (HDFS)

HDFS is Hadoop's primary storage layer, delivering high capacity, high throughput, fault tolerance and cost-effective storage for large datasets through a master–slave architecture comprising one NameNode and multiple DataNodes [14].

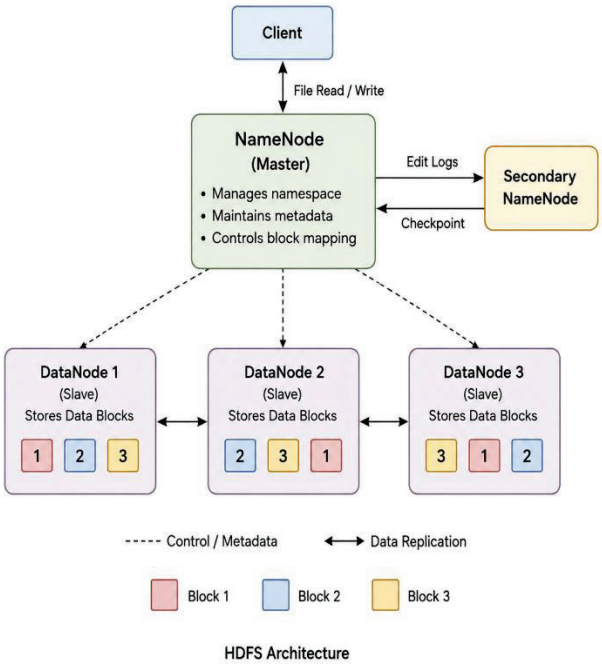

Fig. 2. HDFS master–slave architecture: NameNode, Secondary NameNode and replicated DataNode blocks.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

The NameNode (master) holds filesystem metadata in the fsimage, cached in memory for fast client access; it directs block placement, instructs DataNodes to perform I/O, and monitors the health of the filesystem, making it memory- and I/O-intensive. DataNodes (slaves) store and serve data blocks, replicate them according to the configured replication factor, and coordinate read/write requests with the NameNode. The Secondary NameNode is not a backup or standby node—a common misconception—but periodically checkpoints the namespace image and merges edit logs to keep NameNode metadata current [14]. Splitting files into replicated blocks distributed across DataNodes lets HDFS survive node failures without downtime, scale horizontally, and handle petabyte-scale datasets.

B. pReduce

MapReduce is Hadoop's processing engine, complementing HDFS storage with distributed computation over (key, value) pairs. Input Splitting and a RecordReader first divide HDFS input into logical splits and convert them into (key, value) pairs—by default, the byte offset and line content for text input. Each Mapper processes its split independently, emitting intermediate (key, value) pairs concurrently across the cluster. The Shuffle-and-Sort phase then groups values by key and orders keys sequentially so a Reducer receives logically structured data; Reducers can begin as soon as sufficient intermediate data is available, rather than waiting for all Map tasks to finish, which improves parallelism. The final Reduce phase aggregates values per key and writes the result back to HDFS. Many MapReduce jobs also define an optional Combiner—a local, per-Mapper mini-reduction that pre-aggregates intermediate output before it is shuffled across the network—which reduces the volume of data transferred between the Map and Reduce phases and is one of the more effective, low-effort optimizations available to MapReduce developers.

Architecturally, MapReduce [13] used a master JobTracker to submit jobs, communicate with the NameNode, distribute work to per-node TaskTracker daemons, and reschedule tasks on heartbeat failure—creating a single point of failure if the JobTracker itself failed. TaskTrackers manage a fixed number of task slots and run multiple JVMs for parallel execution [14]. Extensions such as G-Hadoop generalize this model across multiple High-End-Computing clusters connected by high-speed networks, replacing local HDFS with the Gfarm global file system for wide-area, GPU-accelerated data-intensive computing while preserving the Hadoop API [9].

C. Y N (Yet Another Resource Negotiator)

In 2012 MapReduce was split into MapReduce and YARN, separating resource management from job scheduling to improve scalability and efficiency [16]. YARN's Resource Manager coordinates cluster-wide memory, disk, network and CPU allocation via a Scheduler (First-Come-First-Serve, Fair Share or Capacity Scheduler, with First-Come-First-Serve as the default) and an Application Manager, which handles job submission, container assignment and Application Master recovery. Node Managers (replacing TaskTrackers) run on each slave, while the Application Master (replacing the JobTracker) requests resources from the Resource Manager on behalf of a running job through containers—bundles of CPU and memory. By decoupling scheduling from execution, YARN removed the JobTracker bottleneck and let Hadoop clusters run non-MapReduce frameworks such as Spark and Flink alongside batch jobs. A typical application lifecycle begins when a client submits a job to the Resource Manager, which allocates a container for that application's Application Master; the Application Master then negotiates further containers from the Resource Manager for the individual tasks, monitors their progress through the corresponding Node Managers, and requests replacement containers if a task fails—localizing failure recovery to the application level rather than requiring a cluster-wide restart.

D. Hadoop Common

Hadoop Common is the shared set of Java libraries, APIs and utilities—configuration, serialization, RPC, security and I/O—that underpin HDFS, YARN and MapReduce and let ecosystem components such as Hive, Pig, HBase and Oozie interoperate on a common runtime, logging and metrics foundation.

IV. H OP ECOSYSTEM

As Hadoop matured, an ecosystem of Apache-governed tools emerged around it to extend usability and functionality—spanning data storage and query, processing engines, data ingestion, coordination and serialization [13]. Table II summarizes the principal components; all share the same ASF governance and Apache License 2.0, which keeps them interoperable and vendor-neutral.

TABLE II
PRINCIPAL HADOOP ECOSYSTEM COMPONENTS

Tool	Role in the Hadoop Ecosystem
Hive	SQL-like data-warehouse layer (HiveQL) that compiles queries to MapReduce, Tez or Spark jobs; organizes data into databases, tables, partitions and buckets on HDFS.
Pig	Scripting platform (Pig Latin) that expresses multi-step ETL and transformation pipelines as data flows, compiled into MapReduce jobs; runs in local or MapReduce mode.
HBase	Column-oriented, Bigtable-style NoSQL store on top of HDFS for real-time random read/write access to sparse, billion-row tables.
Oozie	Java-based workflow scheduler that chains MapReduce, Pig, Hive, Sqoop and shell actions into DAG workflows, coordinator jobs and bundles using XML-based hPDL.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Tool	Role in the Hadoop Ecosystem
Sqoop	Bulk-transfer utility that moves data between relational databases/warehouses and HDFS, Hive or HBase using Map-only MapReduce jobs and JDBC connectors.
ZooKeeper	Centralized coordination service exposing a hierarchical znode namespace, sessions and watches for leader election, locking and configuration management.
Mahout	Scalable machine-learning library (classification, clustering, collaborative filtering) originally on MapReduce, now supporting Spark and H2O back-ends.
Spark	In-memory batch/stream engine built around Resilient Distributed Datasets (RDDs) and DAG scheduling; substantially faster than MapReduce, with MLlib and GraphX libraries.
Flume	Distributed log-collection service that streams events from sources through channels to sinks (typically HDFS) with tunable reliability.
HCatalog	Table and storage-management layer over the Hive metastore that gives Pig, MapReduce and other tools a shared, format-agnostic view of table schemas.

Among these, Hive and Pig lower the barrier to entry for SQL-oriented and scripting-oriented developers respectively, while HBase, Sqoop and Flume address real-time access and continuous data ingestion that batch-oriented HDFS and MapReduce alone cannot provide. ZooKeeper's hierarchical znode namespace, ephemeral and sequential nodes, and one-time “watch” notifications provide the leader-election and configuration-coordination primitives that HBase, Oozie and other distributed services build upon [6]. Spark's in-memory RDD model and DAG scheduling have made it a dominant processing engine for iterative and interactive workloads, often replacing MapReduce as the execution layer beneath Hive and Pig.

Hive itself now supports several execution back-ends with different performance profiles: Hive-on-Spark replaces MapReduce with Spark for large batch queries; Hive-on-Tez targets large, petabyte-scale batch processing where near-sub-second response is not required; and Hive-on-LLAP keeps persistent, in-memory-caching query servers running so that smaller, interactive queries return in sub-second time. This range of execution modes lets a single HiveQL interface serve ETL, ad-hoc exploration and near-real-time dashboards from the same warehouse layer, which is one reason Hive remains the most common SQL entry point into Hadoop-resident data despite lacking native support for row-level updates or OLTP-style transactions.

V. HA OP SECURITY LANDSCAPE

Hadoop security is conventionally assessed against the CIA triad—confidentiality, integrity and availability [16]. Hadoop was originally designed for trusted, low-security environments; Kerberos was later added for perimeter authentication without extending controls inside the cluster itself. Subsequent projects—Apache Sentry, Apache Knox, Apache Ranger and Project Rhino—along with Kerberos integration with Active Directory and LDAP, added further layers, but none unify into a single, centralized Hadoop security architecture; each solves a narrower problem independently [16].

Key vulnerability classes include identification and authorization gaps, addressed partially through AD/LDAP-Kerberos integration and role-based access control; access-control weaknesses arising from default and inconsistently patched configurations; the absence of built-in monitoring and auditing to detect misuse; and insecure web APIs that expose Hadoop-backed applications to common attacks such as buffer overflow and command injection [16].

These structural weaknesses remain exploitable. Automated cryptomining botnets—including campaigns attributed to Kinsing, the 8220 Gang and TeamTNT-linked actors—have continued through 2024–2026 to target Hadoop clusters left with default configurations, no authentication and exposed management ports such as 8088, underscoring that Hadoop still lacks a single, default-secure architecture years after these gaps were first documented.

Practical mitigation therefore falls to the operator rather than the framework: enabling Kerberos authentication across every service rather than only at the perimeter, firewalling or disabling public exposure of administrative web UIs and REST endpoints (ResourceManager, NameNode and DataNode ports among them), applying role-based access control consistently through tools such as Apache Ranger or Sentry, and maintaining current patch levels across all ecosystem components rather than only the core HDFS/YARN/MapReduce stack. Because these controls are optional add-ons rather than defaults, clusters that are provisioned quickly for development or proof-of-concept use—and then left running—remain the most common entry point for the automated attacks noted above.

VI. APP ATIONS IN INDUSTRY

Hadoop underpins data-driven systems across nearly every major industry because it stores and processes structured, semi-structured and unstructured data cost-effectively at scale. In transportation, Hadoop-based analytics mine fixed-detection, floating-vehicle, GPS and smartphone-detection data for real-time traffic monitoring and prediction, infrastructure monitoring, routing optimization and license-plate-based surveillance. Healthcare institutions use Hadoop-based platforms for predictive diagnostics, outbreak detection and personalized medicine; financial institutions apply it to fraud detection, risk modelling, credit scoring and regulatory reporting; and insurers use it to identify fraudulent claims.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Retailers rely on Hadoop for recommendation engines, market-basket analysis, inventory optimization, dynamic pricing and sentiment analysis, while telecom operators analyze call-detail records and network logs to optimize performance and predict churn. Utilities process smart-meter and IoT data for predictive maintenance and demand forecasting, oil and gas firms analyze seismic data for exploration, and governments apply Hadoop to citizen-data management, welfare-fraud detection and cybersecurity analytics. Search engines, media platforms and manufacturers likewise use Hadoop for indexing, content recommendation and sensor-driven quality control, and research institutions apply it to climate modelling, astronomy and bioinformatics—making Hadoop, through HDFS for storage and MapReduce/YARN for parallel processing, a backbone technology for large-scale, economically viable analysis across sectors.

These use cases have more in common than the common data format: each creates structured, semi-structured or unstructured data at a volume and velocity that a single server cannot economically store or process. Hadoop's value across such varied domains therefore comes less from any one algorithm than from its ability to keep storage and compute costs roughly linear in data volume, which is what makes previously impractical, large-scale analysis economically viable in sector after sector.

VII. ADV NCED ANALYTICS AND TRENDS

Advances in big-data technology now let researchers and organizations produce, store, retrieve and analyze vast quantities of experimental and operational data quickly and at relatively low cost, which has made big-data techniques commonplace across many research domains. Many scientific applications have adopted the MapReduce programming model specifically to take advantage of Hadoop's combined storage and computing power, moving well beyond Hadoop's original web-indexing use case.

Beyond storage and batch processing, Hadoop has matured into a platform for advanced analytics. Research surveys report Hadoop and MapReduce being applied to large-scale genomics and medical-data analysis in bioinformatics—fault-tolerant, scalable distributed storage and parallel processing being well suited to terabyte-scale biological datasets—and to weather-forecasting and geospatial workflows, where MapReduce-based frameworks have been used to model temperature distributions and to add spatial support to social-media-derived location data, with Hadoop clusters shown to outperform single-machine implementations of equivalent processing pipelines.

Data mining on Hadoop, via Mahout and Spark MLlib, applies clustering, association-rule mining and classification to petabyte-scale HDFS data to surface purchasing patterns, fraud signals and operational inefficiencies.

Machine learning workloads distribute model training for classification, regression, clustering and recommendation across commodity clusters using Mahout, Spark MLlib and H2O, shortening training time on billion-record datasets.

Natural language processing tasks—sentiment analysis, feedback mining and document classification—exploit MapReduce or Spark parallelism to tokenize, tag and classify large unstructured text collections.

Vectorization techniques such as TF-IDF and Word2Vec convert unstructured text into numerical representations at scale, enabling downstream similarity search, recommendation and clustering; deep-learning-based embeddings extend this to documents, images and user behaviour.

Predictive analytics layers machine learning frameworks on top of Hadoop's ability to ingest data over the long haul and predicts churn, equipment failure, demand and risk.

Additional domains present more of how Hadoop's scope of analysis is growing.

Similar **graph analytics** frameworks, including Apache Giraph and Spark GraphX, run on graph data that is stored on HDFS, enabling social-network analysis, fraud-ring detection in financial services, user–item recommendation graphs and supply-chain optimization at scales that are not possible for single-machine graph libraries.

While not originally part of Hadoop's batch-oriented design, **real-time streaming analytics** has been added on through Apache Kafka, Spark Streaming, Apache Flink and Apache Storm, enabling real-time dashboards, point-of-transaction fraud detection and continuous IoT-sensor monitoring alongside traditional batch jobs on the same cluster.

Deep-learning integration enables access to the Hadoop-resident datasets, such as TensorFlow (through TensorFlowOnSpark), PyTorch and H2O.ai, enabling the training of convolutional and recurrent neural networks directly against HDFS-stored data and providing a bridge between the traditional big data infrastructure and the latest AI pipelines.

TABLE III

ADVANCED-ANALYTICS DOMAINS AND REPRESENTATIVE HADOOP-ECOSYSTEM TOOLING

Domain	Representative Tools	Typical Use Case
Data mining	Mahout, Spark MLlib	Clustering, association-rule mining, classification on HDFS data
Machine learning	Mahout, Spark MLlib, H2O	Distributed model training for classification/regression/recommendation
NLP	MapReduce, Spark	Sentiment analysis, feedback mining, document classification

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Domain	Representative Tools	Typical Use Case
Vectorization	Spark, MapReduce	TF-IDF, Word2Vec and deep embeddings for similarity search
Predictive analytics	Mahout, Spark MLlib	Churn, failure, demand and risk forecasting
Graph analytics	Apache Giraph, Spark GraphX	Social-network, fraud-ring and supply-chain analysis
Streaming analytics	Kafka, Spark Streaming, Flink, Storm	Real-time dashboards, point-of-transaction fraud detection
Deep learning	TensorFlowOnSpark, PyTorch, H2O.ai	CNN/RNN training on HDFS-resident datasets

VIII. ESEARCH GAP, CHALLENGES AND FUTURE DIRECTIONS

The current literature is sparse and disjointed – the majority of studies focus on an individual tool, algorithm and/or use case, and fail to place its focus in the historical and architectural context of Hadoop. If you are new to the field, you have to assemble the history, naming and evolution of the framework from a multitude of sources. A parallel gap exists in advanced-analytics research—individual studies address data mining, machine learning, NLP, vectorization or predictive analytics separately, but few provide integrative or comparative evaluation of Hadoop's capability and limitations across these subdomains, or benchmark it against non-Hadoop, serverless or Spark-only alternatives.

Persistent technical challenges include HDFS's inefficient random-read performance, a consequence of its batch-oriented design bias; a Hadoop security ecosystem that remains a bolt-on collection of independent tools (Kerberos, Ranger, Sentry, Knox, Rhino) rather than a unified, default-secure architecture increasingly targeted by automated threats; limited empirical, as opposed to conceptual, evaluation of Hadoop's advanced-analytics performance; and energy-efficiency methods that are typically single-purpose rather than adaptive across heterogeneous hardware [16]. Another operational challenge is the lack of reproducibility; published Hadoop research tends to be measured on the cluster configuration and data set size and tuning parameters that are not always fully reported, making it difficult to replicate the reported performance and thus difficult to transfer research results to production systems.

A. Future Directions

Based on these gaps, a number of priorities for the future arise. First, a single source that brings together the past, names and technical architecture of Hadoop in one place will bring down the barrier to entry for new users and help minimize reliance on diverse, tool-specific sources, which this review has sought to squarely tackle head-on. Second, there is a need for dedicated, empirically-rich studies to compare the performance of Hadoop against alternatives, such as non-Hadoop, serverless and Spark-only solutions, for data mining, machine learning, NLP, vectorization and predictive-analytics workloads, thus ensuring that claims about Hadoop's suitability for advanced analytics is backed by measured evidence, not just conceptual discussion.

Third, security research needs to be shifted from disjointed and supplemental tools to centralized and default secure architectures that have monitoring and auditing built in, instead of having to put together the right combination of Kerberos, Ranger, Sentry, and Knox by hand. Fourth, research on energy consumption should be encouraged to support adaptive, multi-objective, machine-learning-based optimization techniques that can cope with the heterogeneity of the cluster hardware, instead of the static, single-objective optimization approach used in most current research. Fifth, the ever closer linkage of deep-learning platforms (TensorFlow, PyTorch) with big-data systems would facilitate the integration of Hadoop's distributed storage and processing capabilities with the deep-learning systems, allowing for large-scale AI training pipelines without a data-movement component. Finally, there is a need for greater interaction between academic research and production Hadoop deployments, such as full disclosure of cluster configurations and tuning parameters in published studies, to promote reproducibility and enhance the ability to effectively implement proposed innovations.

IX. D USSION

Combined, the results of this review form a clear path from scalable and fault-tolerant distributed storage to a platform that can be used to power the intelligence-driven analytics. HDFS and MapReduce, the two fundamental components of the Hadoop system designed for processing huge amounts of data on commodity, low-cost hardware, continue to be the driving force of Hadoop's continued relevance, and the introduction of YARN was a major paradigm shift that took the resources management away from job scheduling and enabled the sharing of the same cluster by other frameworks, including Spark and Flink.

This core is followed by an ecosystem of its own, which is equally important. Hive and Pig made the introduction of SQL and Script-based developers easier, whereas HBase, Sqoop and Flume solved real-time access and continuous data intake challenges in practice. This ecosystem-driven growth is also what has led to the fragmentation this paper identifies as the key research gap: individual studies have tended to focus on a single tool without placing it in the historical, architectural and ecosystem context of Hadoop,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

which is something that newcomers to the ecosystem must then put together themselves.

While security has been added in via Kerberos, Apache Ranger and Apache Sentry, the repeated security theme from Section V—no single, centralized security architecture—is a testament to the fact that security research and tooling has not been able to catch up with this rapidly expanding attack surface, which remains vulnerable to automated cryptomining campaigns through 2024–2026. Finally, while one of the most frequently cited use cases of Hadoop in industry is in the realm of advanced analytics, much of the literature reviewed in Section VII focuses on data mining, machine learning, NLP, vectorization and predictive analytics in a general sense, and rigorous, Hadoop-specific benchmarking of performance, scalability and limitations in these subdomains is comparatively limited.

X. CONCLUSION

This review documents Hadoop's journey from the Apache Nutch project to a fully-fledged platform for distributed storage, processing and advanced analytics, and brings together the history of Hadoop, its architecture (HDFS, MapReduce, YARN and Hadoop Common), the tools of the ecosystem, its security situation and its industry applications in a structured reference. It also outlines the development of Hadoop in the field of data mining, machine learning, natural language processing, vectorization, predictive analytics and further highlights a persistent lack of integrative, empirical research in these subdomains. Future research efforts should focus on rigorous performance benchmarking for Hadoop's advanced-analytics performance, on providing a unified default-secure architecture to withstand automated attacks, and on increased coupling between research and production deployments, for both new players interested in fundamental understanding and existing players interested in more advanced and modern applications of Hadoop.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” in Proc. 19th ACM Symp. Operating Systems Principles (SOSP '03), Bolton Landing, NY, USA, Oct. 2003, pp. 29–43.

[2] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” in Proc. 6th Symp. Operating Systems Design and Implementation (OSDI '04), San Francisco, CA, USA, Dec. 2004, pp. 137–150.

[3]] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: a distributed storage system for structured data,” in Proc. 7th Symp. Operating Systems Design and Implementation (OSDI '06), Seattle, WA, USA, Nov. 2006, pp. 205–218.

[4] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, “Pig Latin: a not-so-foreign language for data processing,” in Proc. 2008 ACM SIGMOD Int. Conf. Management of Data, Vancouver, BC, Canada, Jun. 2008, pp. 1099–1110.

[5] A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, “Hive — a warehousing solution over a map-reduce framework,” Proc. VLDB Endowment, vol. 2, no. 2, pp. 1626–1629, 2009.

[6] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, “ZooKeeper: wait-free coordination for Internet-scale systems,” in Proc. 2010 USENIX Annual Technical Conference (USENIX ATC '10), Boston, MA, USA, Jun. 2010.

[7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop Distributed File System,” in Proc. 2010 IEEE 26th Symp. Mass Storage Systems and Technologies (MSST), Incline Village, NV, USA, May 2010, pp. 1–10.

[8] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: cluster computing with working sets,” in Proc. 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10), Boston, MA, USA, Jun. 2010.

[9] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen, “G-Hadoop: MapReduce across distributed data centers for data-intensive computing,” Future Generation Computer Systems, vol. 29, no. 3, pp. 739–750, 2013.

[10] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, “Apache Hadoop YARN: yet another resource negotiator,” in Proc. 4th Annual Symp. Cloud Computing (SoCC '13), Santa Clara, CA, USA, Oct. 2013, Art. no. 5, pp. 1–16.

[11] A. O'Driscoll, J. Daugelaite, and R. D. Sleator, “Big data, Hadoop and cloud computing in genomics,” Journal of Biomedical Informatics, vol. 46, no. 5, pp. 774–781, 2013.

[12] M. Chen, S. Mao, and Y. Liu, “Big data: a survey,” Mobile Networks and Applications, vol. 19, no. 2, pp. 171–209, 2014.

[13] C. Uzunkaya, T. Ensari, and Y. Kavurucu, “Hadoop ecosystem and its analysis on tweets,” Procedia - Social and Behavioral Sciences, vol. 195, pp. 1890–1897, 2015.

[14] M. R. Ghazi and D. Gangodkar, “Hadoop, MapReduce and HDFS: a developers' perspective,” Procedia Computer Science, vol. 48, pp. 45–50, 2015.

[15] The Apache Software Foundation, “The Apache Software Foundation announces Apache Hadoop v3.0.0 general availability,” ASF Blog, news.apache.org, Dec. 2017.

[16] G. S. Bhathal and A. Singh, “Big data: Hadoop framework vulnerabilities, security issues and attacks,” Array, vol. 1, art. 100002, 2019.