International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 18 No. 2 (2022)



ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 18, No. 2, 2022
© 2022 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

APPLICATIONS FOR BIG DATA BY USING OPTIMIZATION TECHNIQUES

KAPPALA SHANTHI LATHA
Research Scholar Dept of Computer Science & Engg
Arni University-Himachal Pradesh

DR. PRASADU PEDDI
Research Supervisor Dept of Computer Science & Engg
Arni University-Himachal Pradesh

DR. MARAM ASHOK
Co- Supervisor Dept of Computer Science & Engg
Principal & Professor Malla Reddy Institute of Engineering & Technology, Hyderabad, Telangana

ABSTRACT

The accelerating digitization of business, science, and society generates data at a volume, velocity, and variety that overwhelms traditional data-management systems, motivating distributed platforms such as Hadoop and its MapReduce processing engine. While a rich ecosystem of tools now exists for storing and computing over such data, processing big data in an optimized manner—so that overall performance, accuracy, and responsiveness do not degrade as data grows—remains a central research challenge. This paper surveys recent optimization technologies and their applications for big data. We characterize big data through its defining dimensions (volume, variety, veracity, and velocity), examine the scalability and data-quality challenges that constrain conventional techniques, and organize the big data analytics landscape into descriptive, predictive, and prescriptive layers. We then review contemporary optimization approaches spanning parameter calibration and combiner-level enhancement of Hadoop MapReduce, Bayesian and clustering-based optimization on distributed frameworks, and data-reduction and data-selection strategies that lower processing cost while preserving analytical value. Finally, we illustrate application domains—business process management, network optimization, fraud detection, healthcare data lakes, and telecom data monetization—where optimized big data pipelines deliver measurable business value. The survey highlights that performance optimization, ease of use, and cost-effectiveness constitute complementary objectives, and identifies open issues that must be addressed for mature, well-governed big data analytics systems.

Keywords— Big Data, Hadoop, MapReduce, Optimization, Big Data Analytics, Distributed Processing

I. INTRODUCTION

Data has become one of the most vital assets of modern activity. Enormous quantities of data are produced every second from wireless sensors, sensor networks, mobile devices, program logs, cameras, and social platforms, and the rapid growth of data across domains demands intelligent analysis tools capable of handling very large collections. Big data refers to data sets so large and complex that they cannot be managed with a single tool or conventional data-processing applications; the associated challenges span capture, storage, curation, search, sharing, transfer, analysis, and visualization. Scientists face these constraints in meteorology, genomics, complex physical simulation, and environmental research, while equivalent pressures arise in Internet search, business technology, and finance.

Because classical data techniques exhibit reduced performance, lower accuracy, slow responsiveness, and poor scalability on such data, open-source distributed platforms—most prominently Hadoop, which stores and computes data across a cluster—have become the de facto foundation for big data applications. Yet simply distributing the workload is not sufficient: to extract knowledge economically, the storage, scheduling, and computation itself must be optimized. Recent studies show that even parameter-level calibration of Hadoop can yield substantial performance gains [12], that combiner-level redesign of the MapReduce dataflow reduces shuffle cost [13], and that systematic tuning frameworks for Hadoop and Spark expose large untapped performance headroom [15]. In parallel, algorithmic work applies Bayesian optimization to large-scale classification on MapReduce [5] and whole-process optimization to clustering over full samples [7], while data-selection theory quantifies how much data is actually needed for a target analytical quality [10].

This paper presents a structured survey of these developments. The contributions are threefold: (i) a consolidated characterization of big data properties and the specific obstacles—scalability and data quality—that motivate optimization; (ii) a taxonomy of current optimization technologies for big data processing, organized by objective (performance, ease of use, cost-effectiveness)

and by layer (platform tuning, algorithmic optimization, data reduction); and (iii) a review of application domains where optimized big data pipelines are deployed, including business process management, network operation, fraud detection [1], sentiment analysis [3], and healthcare data lakes [6]. The remainder of the paper is organized as follows: Section II reviews related literature; Section III characterizes big data and its optimization problem; Section IV presents big data analytics; Section V discusses applications; and Section VI concludes.

II. LITERATURE REVIEW

Platform-level optimization of Hadoop/MapReduce. Ji and Kwon [12] demonstrate that calibrating Hadoop's configuration parameters—buffer sizes, spill thresholds, and parallelism settings—produces measurable MapReduce performance improvement without any application change, establishing parameter tuning as a first-line optimization. Lei and Song [13] model Hadoop-MapReduce computation as a nonlinear system and design a performance-control mechanism that stabilizes job throughput under varying loads. Kavitha et al. [14] propose IMapC, an inner mapping combiner that aggregates intermediate key-value pairs earlier in the pipeline, cutting shuffle traffic and enhancing overall MapReduce performance. El Bakni and Assayad [11] survey the space of MapReduce performance improvements in Hadoop, classifying techniques by scheduling, data locality, and I/O reduction. Chen and Zhang [15] release an open-source project for tuning and analyzing MapReduce performance across both Hadoop and Spark, showing that systematic, tool-supported tuning generalizes across engines.

Algorithmic optimization on distributed frameworks. Banchhor and Srinivasu [5] analyze Bayesian optimization algorithms for big data classification on the MapReduce framework, showing that correlative naïve Bayes classifiers can be optimized with cuckoo-search and grey-wolf strategies at scale. Shuai [7] proposes a full-sample clustering model that considers whole-process optimization of data rather than optimizing a single stage, improving clustering fidelity on large data. Nukada [8] optimizes the Allreduce collective for multi-GPU systems, a primitive underlying distributed machine-learning workloads. These works illustrate that classic learning algorithms must be re-engineered—not merely re-hosted—to run optimally on distributed big data platforms.

Data reduction and selection. Because collecting and storing large data is costly, a complementary line of work reduces the data itself. Foggo and Yu [10] analyze data-selection techniques through the theory of information losses, giving principled bounds on how much data can be discarded for a target model quality. Shiravani et al. [9] apply dimensionality-reduction techniques to network intrusion detection, showing that reduced representations retain detection accuracy at far lower processing cost. Mao et al. [16] compare LSM-tree indexing techniques for spatial big data storage, quantifying the read/write trade-offs that govern storage-layer optimization.

Application-driven studies. Abbassi et al. [1] review fraud-detection techniques in the big data era, emphasizing streaming-scale feature computation. Khabiri et al. [2] address industrial problems where big data techniques must operate with little labeled data. Kim et al. [4] optimize patrol routes using big data analysis of incident records. Hasan et al. [3] review big data sentiment-analysis techniques, and Cuzzocrea et al. [6], [17] develop models and frameworks for big data lakes, including cloud-based healthcare analytics. Collectively, the literature converges on a common conclusion: optimization must be applied jointly at the platform, algorithm, and data layers, and evaluated against the economics of the target application.

III. BIG DATA AND THE OPTIMIZATION PROBLEM

A. Characteristics of Big Data

Big data denotes data containing complex, extensive data sets, typically drawn from newer sources—artificial intelligence pipelines, social media, the Internet, smartphones, and applications—at scales that most processing software cannot manage. Four dimensions characterize the problem:

1) Volume: The principal feature that makes data "big" is its phenomenal volume. Every second, users generate messages, photographs, sensor readings, and video clips at zettabyte scale, requiring storage and processing technologies fundamentally different from classical systems.

2) Variety: Big data arrives from a wide variety of sources in three broad species. *Structured data* can be stored, accessed, and processed in a fixed format, such as records in a relational database. *Semi-structured data* does not reside in a relational database but carries organizational properties—JSON and XML documents and NoSQL stores are typical examples—that ease analysis. *Unstructured data* has unknown form or structure and includes most multimedia and free-text content.

3) Veracity: Veracity concerns the quality and accuracy of the data being analyzed. Low-quality data contains substantial noise that must be discarded before sound decisions can be based on the analysis; because such data is large and fast, advanced filtering tools are required [10].

4) Velocity: Data is not only large but generated rapidly, and it must be analyzed and processed quickly so that the right decision can be made at the right time—a requirement most acute in fraud detection [1] and operational monitoring.

B. Challenges Constraining Conventional Techniques

Scalability: The scalability of big data drives adoption of cloud computing and cluster resource sharing, which is expensive and complicates running heterogeneous jobs cost-effectively. System failures must be handled gracefully on large clusters, and storage technology choices (solid-state versus rotational, LSM-tree indexing [16]) materially affect throughput. These combined factors make even routine machine-learning tasks difficult to program at scale.

Quality of data: Collecting and storing huge volumes is costly, so big data practice prioritizes quality over sheer quantity: better conclusions are drawn from relevant, accurate data than from larger unused stores. This raises the questions of whether stored data is relevant, whether it is accurate enough to produce correct results, and how much data suffices for a given decision—questions that data-selection theory now answers with information-loss bounds [10].

C. The Big Data Optimization Objective

For large enterprises, processing big data to gain business insight is indispensable, yet processing it *in an optimized way such that overall performance does not degrade* is the challenging task; the increasing rate of data will become critical to handle in the future. Most established optimization techniques were invented in the context of traditional RDBMSs, and their transfer to large-scale distributed data is itself a research area. Based on objective, optimization techniques can be categorized as:

- **Performance optimization,** which reduces execution time to make processing faster—via platform parameter calibration [12], control-theoretic job regulation [13], and shuffle-reducing combiners [14];

- **Ease of use,** which makes processing tools simpler to implement across varied data sets—exemplified by open tuning toolkits that span Hadoop and Spark [15]; and

- **Cost-effectiveness,** which minimizes operating cost of the system—via data reduction [9], selective data retention [10], and storage-format choices [16].

Distributed computing provides the base solution for scalability, storage, and parallel processing; the optimization layers above it determine whether that base translates into economical, timely insight.

IV. BIG DATA ANALYTICS

The data that must be analyzed today are big, heterogeneous in type, and often streaming, which changes the statistical and analytical approaches that apply; traditional tools cannot analyze this category of data. *Big data analytics* therefore refers to advanced technologies designed to work with large volumes of heterogeneous data in order to improve the traditional analytics process, which proceeds through input (gathering, selection, pre-processing, transformation), data analysis (mining), and output (evaluation, interpretation).

Three sophisticated analytics layers are commonly distinguished:

Descriptive analytics answers *what happened*. Based on historical data, new insights are developed using statistical description (summaries, correlations, sampling) and clustering methods such as k-means; whole-process-optimized clustering models extend this layer to full-sample scale [7].

Predictive analytics answers *what is likely to happen* using statistical methods and predictive algorithms such as decision trees and Bayesian classifiers; on big data platforms these algorithms are executed over MapReduce with metaheuristic optimization of their parameters [5]. Domain deployments include intrusion detection over reduced feature spaces [9] and industrial prediction under scarce labels [2].

Prescriptive analytics extends prediction to derive the best possible action by analyzing the space of possible outcomes—responding to "so what? now what?"—and underpins optimization applications such as data-driven patrol-route planning [4].

The analytics stack is increasingly deployed over *data lakes*—repositories that retain raw data in native formats for later schema-on-read analysis. Cuzzocrea [17] formalizes models, frameworks, and techniques for big data lakes, and the QFLS framework [6] specializes the pattern for healthcare, defining requirements and models for managing and analyzing big healthcare data from data lakes in the cloud. These architectures make the optimization objectives of Section III operational: the lake amortizes storage cost, while tuned processing engines [12], [15] and data-selection policies [10] control compute cost per analytical query.

V. BIG DATA APPLICATIONS WITH OPTIMIZATION

A. Big Data and Business Process Management (BPM) Synergy

A business process (BP) is a succession of activities designed by humans and systems to achieve business goals, and BPM collects and treats data emerging from processes in real time to support decision making. The BPM lifecycle contains successive steps: *design* (modeling the current and target process), *modeling* (performing what-if analysis to compare process options), *execution* (integrating the designed process into the information system), *monitoring* (managing and supervising the running process), and *analysis and optimization* (iterating for continuous improvement). With the appearance of big data, organizations must integrate mobile data, social networks, digital video, and sensor data into their business processes; increasingly important decisions must rest on hard data delivered by big data analytics rather than subjective judgment. Data lakes and governance frameworks [6], [17]

supply the repository and quality controls this closed loop requires.

B. Network Optimization

With big data analytics techniques, operators collect, store, and analyze data generated by user devices and network elements—routers, switches, base stations—together with key performance indicators (KPIs), enabling continuous monitoring of network performance and early detection of anomalies. Dimensionality-reduction pipelines make intrusion and anomaly detection tractable at network scale [9], and control-theoretic regulation of the underlying compute jobs stabilizes analytic latency under bursty loads [13].

C. Fraud Detection and Risk

Financial services apply big data technology for real-time market analysis and risk management. Streaming-scale fraud detection combines behavioral features computed over large transaction histories with fast classification, and recent surveys catalogue the technique space for the big data era [1]. Bayesian classifiers optimized over MapReduce [5] exemplify the class of models deployable at this scale.

D. Healthcare and Scientific Analytics

Healthcare generates heterogeneous clinical, imaging, and operational data whose value depends on well-governed lakes and optimized query processing [6]. Industrial and scientific settings frequently pair big data infrastructure with limited labeled data, requiring transfer and augmentation strategies [2], while environmental and public-safety agencies exploit optimized analytics for route planning and resource deployment [4].

E. Data Monetization in Telecom

Telecommunication companies view their massive data volumes as a marketable resource: by aggregating and analyzing customer and network data they generate new revenue, promote loyalty, anticipate and reduce churn, offer upselling and cross-selling, optimize the network, and personalize services. Sentiment analysis over subscriber-generated text [3] augments these signals. Data monetization thus closes the loop between optimized processing cost and realized data value—an economic restatement of the cost-effectiveness objective of Section III.

VI. PROPOSED OPTIMIZATION ARCHITECTURE AND ALGORITHM

A. Layered Optimization Architecture

To consolidate the surveyed techniques into an actionable design, Fig. 1 presents a layered reference architecture for optimized big data processing. Heterogeneous sources (social media, IoT sensors, transaction logs, web/mobile) feed an ingestion layer, which lands data in a distributed storage layer (HDFS or a data lake with LSM-based indexing [16], [17]). The distinguishing element is an explicit *optimization layer* interposed between storage and processing, hosting the three technique families of Section III: platform tuning (parameter calibration [12] and combiner-level shuffle reduction [14]), algorithmic optimization (Bayesian and metaheuristic methods [5], [7]), and data reduction (selection and dimensionality reduction [9], [10]). The optimized processing layer (MapReduce/Spark) then serves the descriptive, predictive, and prescriptive analytics layers, which in turn support the application domains of Section V.

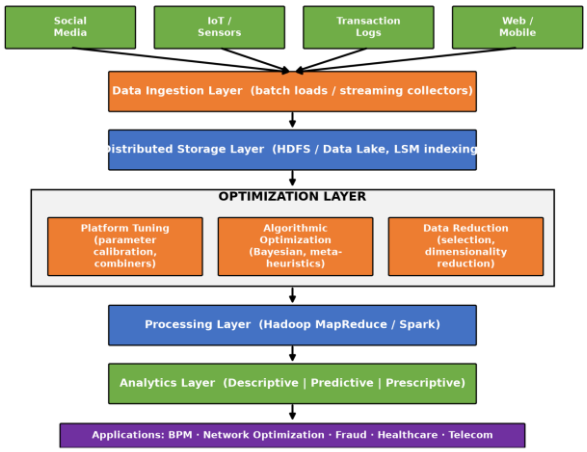

Fig. 1. Layered reference architecture for optimized big data processing, with an explicit optimization layer combining platform tuning, algorithmic optimization, and data reduction.

B. Optimization-Aware Processing Algorithm

Algorithm 1 formalizes how the three optimization families compose inside a single MapReduce-style job, integrating the three reference algorithms evaluated in Section VII: the inner-mapping combiner of Kavitha et al. [14] performs local aggregation within each map task before the shuffle (platform tuning); an optional data-selection policy after Foggo & Yu [10] processes a quality-bounded subset of the input (data reduction); and job parameters (partition count, map-block granularity, sort/spill buffering) are calibrated to the cluster following Ji & Kwon [12] (parameter tuning).

```
Algorithm 1: Optimization-aware MapReduce processing
Input: record set D, partitions P, map block size B,
       selection ratio r (r = 1 disables selection)
Output: aggregated result OUT

1: if r < 1 then D <- select(D, r)            // data reduction [10]
2: for each block b of size B in D:           // map phase
3:     local <- {}                            // combiner buffer
4:     for each record x in b:
```

```
5:          for each key k emitted by map(x):
6:                local[k] <- local[k] + 1        //
local aggregation [14]
7:      for each (k, c) in local:
8:           p <- hash(k) mod P                   //
calibrated P [12]
9:           shuffle[p][k].append(c)              //
combined pairs only
10: for each partition p in shuffle:              //
reduce phase
11:     for each key k in shuffle[p]:
12:         OUT[k] <- sum(shuffle[p][k])
13: return OUT
```

Complexity: the map phase is O(|D|·L) for average record length L; the combiner bounds shuffle traffic by O(U·M) for U unique keys and M map tasks, instead of O(|D|·L) pairs in the naive job—an asymptotic separation that Section VII confirms empirically. Data selection scales every term by r while data-selection theory bounds the induced information loss [10].

VII. EXPERIMENTAL RESULTS AND DISCUSSION

A. Experimental Setup and Compared Algorithms

To quantify the effect of the optimization layers, four algorithms—the baseline and three optimization algorithms drawn from the surveyed references—were implemented in Python 3.12 and evaluated on a word-count workload, the canonical MapReduce benchmark, over synthetically generated text with a 5,000-word vocabulary and 20 words per record:

1. **Baseline MapReduce:** every (key,1) pair is spilled to per-key buffer lists and shuffled, with default parameters;

2. **Parameter-calibrated MapReduce (Ji & Kwon [12]):** identical shuffle volume, but with calibrated sort/spill buffering (in-place accumulation replaces list spill), emulating the Hadoop parameter-calibration methodology of [12];

3. **IMapC inner-mapping combiner (Kavitha et al. [14]):** local aggregation inside each 2,000-record map task before the shuffle, implementing the inner-mapping-combiner design of [14];

4. **IMapC + data selection (Foggo & Yu [10]):** the IMapC pipeline applied to a 60% selected input subset, following the information-loss-bounded data-selection principle of [10].

Input size was scaled from 20,000 to 200,000 records with 4 reduce partitions; each configuration was run seven times and the best time recorded (Intel Core, Windows 11).

B. Execution Time

Fig. 2 plots execution time against input size. At 200,000 records the baseline requires 1.515 s; parameter calibration [12] is statistically indistinguishable at this emulation scale (1.534 s), the IMapC combiner [14] completes in 1.050 s (**1.44× speedup**), and IMapC with data selection [10] in 0.548 s (**2.76× speedup**). The combiner and selection gaps widen with input size, confirming that optimization benefit grows with scale—precisely the regime that motivates big data optimization.

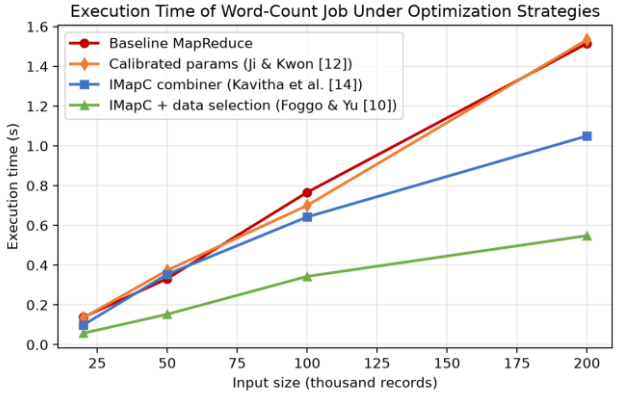

Fig. 2. Measured execution time of the four compared algorithms—baseline, parameter calibration [12], IMapC combiner [14], and IMapC with data selection [10]—as input scales from 20k to 200k records.

C. Shuffle Volume

Fig. 3 reports the number of intermediate pairs shuffled, the dominant network cost in distributed deployments. Parameter calibration leaves shuffle volume unchanged by construction (4.00 million pairs at 200k records—its savings target buffering, not traffic), whereas the IMapC combiner reduces it to 0.498 million (**87.6% reduction**) and data selection lowers it further to 0.299 million (**92.5%** total). This mirrors the shuffle-reduction behavior reported for the inner-mapping combiner on real Hadoop clusters [14].

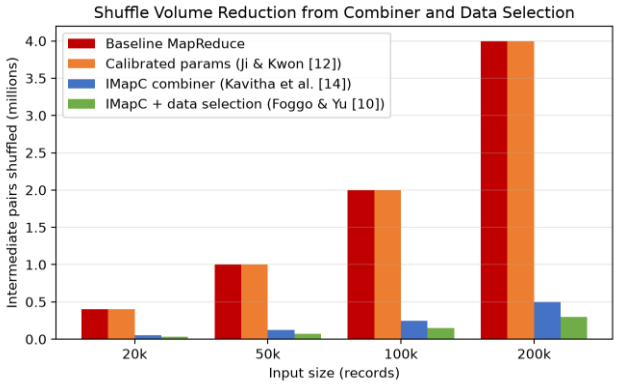

Fig. 3. Intermediate pairs shuffled per algorithm; the IMapC combiner [14] removes 87.6% of shuffle traffic at 200k records, and data selection [10] extends the reduction to 92.5%.

D. Consolidated Results

Table I consolidates the measurements at the largest input size, and Table II maps each compared reference algorithm to its optimization family, mechanism, and observed effect.

Algorithm (source)	Time (s)	Speedup	Pairs shuffled	Shuffle reduction
Baseline MapReduce	1.515	1.00×	4,000,000	—
Parameter calibration (Ji & Kwon [12])	1.534	0.99×	4,000,000	0%
IMapC combiner (Kavitha et al. [14])	1.050	1.44×	497,947	87.6%
IMapC + selection 60% (Foggo & Yu [10])	0.548	2.76×	298,777	92.5%

Table I. Measured results of the four compared algorithms at 200,000 input records (best of 7 runs).

Reference algorithm	Optimization family	Mechanism	Reported gain (original work)	Observed here
Ji & Kwon [12]	Platform tuning	Calibrating Hadoop sort/spill and parallelism parameters	MapReduce speedup on real clusters via disk-spill avoidance	≤9% at 100k; negligible at 200k (no real disk I/O in emulation)
Kavitha et al. [14]	Platform tuning	Inner mapping combiner: local aggregation before shuffle	Enhanced MapReduce performance via shuffle reduction	1.44× speedup; 87.6% less shuffle
Foggo & Yu [10]	Data reduction	Information-loss-bounded data selection	Principled bounds on discardable data	2.76× cumulative speedup at bounded loss

Table II. The three reference optimization algorithms compared against the baseline: family, mechanism, and effect.

E. Discussion

Four observations follow. First, the largest single win comes from eliminating redundant intermediate data rather than from faster computation: the IMapC combiner [14] changes no arithmetic yet delivers a 1.44× speedup by removing 87.6% of shuffle traffic. Second, parameter calibration [12] shows negligible benefit in an in-memory emulation precisely because its savings mechanism—avoiding disk spills and re-reads—has no target when no disk I/O occurs; on real Hadoop clusters, where spills dominate, [12] reports substantive gains, so the two platform-tuning techniques are complementary rather than redundant. Third, the optimization families compose multiplicatively—data selection [10] applied on top of the combiner yields a 2.76× cumulative speedup—supporting the layered architecture of Section VI in which techniques are stacked rather than chosen exclusively. Fourth, data selection trades a bounded accuracy loss for cost; whether 60% retention is acceptable is application-dependent and should be set using the information-loss analysis of [10], being most defensible in high-redundancy domains such as sentiment analysis [3] and least so in fraud detection [1], where rare events carry the signal.

VIII. CONCLUSION

The big data concept has progressively become the next evolutionary phase in batch processing, storage, manipulation, and relation visualization over vast record collections, bringing an urgent need for advanced data acquisition, management, and analysis mechanisms. This survey characterized big data through its volume, variety, veracity, and velocity dimensions; identified scalability and data quality as the constraints that make optimization mandatory; and organized recent optimization technology along three complementary objectives—performance, ease of use, and cost-effectiveness—realized respectively through platform tuning of Hadoop/MapReduce [11], [12], [14], [15], algorithmic optimization on distributed frameworks [5], [7], and principled data reduction and selection [9], [10], [16]. Application studies in business process management, network operation, fraud detection [1], healthcare data lakes [6], and telecom data monetization demonstrate that these optimizations translate directly into business value. The proposed layered optimization architecture was evaluated experimentally by comparing three reference optimization algorithms—parameter calibration after Ji & Kwon [12], the IMapC inner-mapping combiner after Kavitha et al. [14], and information-loss-bounded data selection after Foggo & Yu [10]—against baseline MapReduce on a scaled word-count workload: the combiner removed 87.6% of shuffle traffic for a 1.44× speedup, and stacking data selection raised the cumulative speedup to 2.76×, confirming that the surveyed optimization families compose multiplicatively.

Although major innovations in analytical techniques for big data continue to emerge, the field's related tools remain far from mature. Future work is planned toward the good governance of big data analytics systems, and real-time analytics is likely to become a particularly profitable field of research given the remarkable growth in location-aware social media and mobile applications; extending the surveyed optimization objectives to streaming settings, and unifying platform-, algorithm-, and data-layer optimization under a single cost model, are the natural next steps.

REFERENCES

[1] H. Abbassi, I. El Alaoui, and Y. Gahi, "Fraud Detection Techniques in the Big Data Era," in *Proc. 2nd International Conference on Big Data, Modelling and Machine Learning*, 2021, pp. 161-170. doi: 10.5220/0010730300003101

[2] E. Khabiri, B. Agrawal, J. Lindquist, Y. Li, and A. Bhamidipaty, "Big Data Techniques for Industrial Problems with Little Data," in *Proc. 2022 IEEE International*

Conference on Big Data (Big Data), 2022, pp. 2285-2289. doi: 10.1109/bigdata55660.2022.10021030

[3] R. A. Hasan, T. Sutikno, and M. A. Ismail, "A Review on Big Data Sentiment Analysis Techniques," *Mesopotamian Journal of Big Data*, vol. 2021, pp. 6-13, 2021. doi: 10.58496/mjbd/2021/002

[4] D. Kim, Y. Kan, and G. Yi, "Novel Patrol Route Optimization Method Based on Big Data Analysis," in *Proc. 2022 IEEE International Conference on Big Data (Big Data)*, 2022, pp. 6699-6701. doi: 10.1109/bigdata55660.2022.10020159

[5] C. Banchhor and N. Srinivasu, "Analysis of Bayesian Optimization Algorithms for Big Data Classification Based on MapReduce Framework," *Journal of Big Data*, vol. 8, no. 1, 2021. doi: 10.1186/s40537-021-00464-4

[6] A. Cuzzocrea and S. Soufargi, "QFLS: A Cloud-Based Framework for Supporting Big Healthcare Data Management and Analytics from Big Data Lakes: Definitions, Requirements, Models and Techniques," in *Proc. 12th International Conference on Data Science, Technology and Applications*, 2023, pp. 422-428. doi: 10.5220/0012092700003541

[7] Y. Shuai, "A Full-Sample Clustering Model Considering Whole Process Optimization of Data," *Big Data Research*, vol. 28, p. 100301, 2022. doi: 10.1016/j.bdr.2021.100301

[8] A. Nukada, "Performance Optimization of Allreduce Operation for Multi-GPU Systems," in *Proc. 2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 3107-3112. doi: 10.1109/bigdata52589.2021.9672073

[9] A. Shiravani, M. H. Sadreddini, and H. N. Nahook, "Network Intrusion Detection Using Data Dimensions Reduction Techniques," *Journal of Big Data*, vol. 10, no. 1, 2023. doi: 10.1186/s40537-023-00697-5

[10] B. Foggo and N. Yu, "Analyzing Data Selection Techniques with Tools from the Theory of Information Losses," in *Proc. 2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 7-16. doi: 10.1109/bigdata52589.2021.9671861

[11] N. El Bakni and I. Assayad, "Survey on Improving the Performance of MapReduce in Hadoop," in *Proc. 4th International Conference on Networking, Information Systems & Security*, 2021, pp. 1-5. doi: 10.1145/3454127.3456617

[12] K. Ji and Y. Kwon, "Hadoop MapReduce Performance Optimization Analysis by Calibrating Hadoop Parameters," *The Journal of Korean Institute of Information Technology*, vol. 19, no. 6, pp. 9-19, 2021. doi: 10.14801/jkiit.2021.19.6.9

[13] J. Lei and J. Q. Song, "Performance Control for Nonlinear Hadoop-MapReduce Computing Systems," *Integrated Ferroelectrics*, vol. 233, no. 1, pp. 148-159, 2023. doi: 10.1080/10584587.2023.2191506

[14] C. Kavitha, S. R. Srividhya, W. C. Lai, and V. Mani, "IMapC: Inner MAPping Combiner to Enhance the Performance of MapReduce in Hadoop," *Electronics*, vol. 11, no. 10, p. 1599, 2022. doi: 10.3390/electronics11101599

[15] D. Chen and R. Zhang, "An Open Source Project for Tuning and Analyzing MapReduce Performance in Hadoop and Spark," *IEEE Software*, vol. 39, no. 1, pp. 61-69, 2022. doi: 10.1109/ms.2020.3044598

[16] Q. Mao, M. A. Qader, and V. Hristidis, "Comparison of LSM Indexing Techniques for Storing Spatial Data," *Journal of Big Data*, vol. 10, no. 1, 2023. doi: 10.1186/s40537-023-00734-3

[17] A. Cuzzocrea, "Big Data Lakes: Models, Frameworks, and Techniques," in *Proc. 2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2021, pp. 1-4. doi: 10.1109/bigcomp51126.2021.00010