Int. j. adv. multidisc. res. stud. 2023; 3(6):3040-3046

Received: 10-11-2023
Accepted: 20-12-2023

International Journal of Advanced Multidisciplinary Research and Studies

ISSN: 2583-049X

The Small-Files Problem in Modern Data Lakes: From HDFS NameNode Constraints to Metadata and Manifest Bottlenecks in Iceberg and Delta Lake

Santosh Nagnath Mehtre
Independent Researcher, Data Engineering, United States of America

DOI: https://doi.org/10.62225/2583049X.2023.3.6.6805
Corresponding Author: **Santosh Nagnath Mehtre**

Abstract
The researchers analyze the staying power of the small-files problem in the three popular open-source data lakes: HDFS, Apache Iceberg, and Delta Lake. It benchmarks the effectiveness of compaction in terms of the number of fragments, the number of metadata overhead, the delay of query planning, execution performance and other factors. The results establish that HDFS can result in the most significant degradation over time. Iceberg can have lower overhead over time, and Delta Lake can have lower overhead compared to growth in the transaction log. The effectiveness of compaction is over 98.8% on platforms. The findings indicate that the new table formats don't overcome the challenge of small files, and strategies based on metadata and file optimisation can need to be constantly implemented.

Keywords: Small-Files Problem, HDFS, Apache Iceberg, Delta Lake, Metadata Scalability, Modern Data Lakes

1. Introduction

Background and Research Context

Distributed data platforms have been plagued by a small-files problem that endures. The large number of small files caused a performance problem in the cluster if used in conjunction with HDFS, as all metadata for all files was stored in memory in the NameNode [1]. This architectural dependency is removed with cloud object storage. The other challenges appeared when it came to metadata operations, creating FPs, and latency associated with listing and query-planning [2]. Thus, for the scalability of contemporary table formats, such as Apache Iceberg, Delta Lake, compaction and metadata optimisation and manifest rewrites are required.

Aim and Objectives

Aim

The aim of the research is to learn the current reasons behind the small-files problem in each of these technologies, HDFS, Iceberg, and Delta Lake through metadata bottlenecks and compaction.

Research Objectives

- To examine the effect of historical memory limits on the NameNode on performance problems created by many small files.
- To evaluate bottlenecks with metadata for discovery and query planning in Apache Iceberg data lakes.
- To analyse the hub of inefficiencies of Delta Lake due to small files related through transaction logs, metadata expansion, and partition enumeration.
- To compare different compaction and metadata optimisation strategies for HDFS, Iceberg, and Delta Lake to support scalable analytics.

Problem Statement

Cloud object storage deals with the memory constraints related to HDFS NameNodes and the proliferation of metadata [3]. The growth of metadata manifests, the latency of listing objects, and the overhead of planning queries are all problems with small files [4]. People often discuss them one at a time, leading to a lack of understanding of the impact these would have on the

different system architectures like Hadoop (HDFS), Apache Iceberg, and Delta Lake of the problem.

Novel Contribution

This study brings a comparative perspective and improvements from memory-bound HDFS metadata management to overheads due to cloud-native manifests and transaction logs. Instead of lumping the current data-lake bottlenecks into the modern category, it considers them to be an architectural move from one location to another of the same challenge of scalability [5]. However, it also shows a clear linkage between compaction performance of Iceberg, Delta Lake, HDFS, de-duplication by manifest rewriting, metadata optimisation, and query planner for both HDFS.

2. Literature Review

HDFS NameNode Constraints and the Origins of the Small-Files Problem

HDFS is an original version of small files and has a metadata structure that is centrally managed [6]. The millions of small files consume an undue amount of memory because each file and block requires a metadata entry to be recorded in the memory of the NameNode, and increases the overhead for Namespace management [7]. It does not scale as well as others because it cannot be the storage space constraint that the amount of metadata space is the constraint.

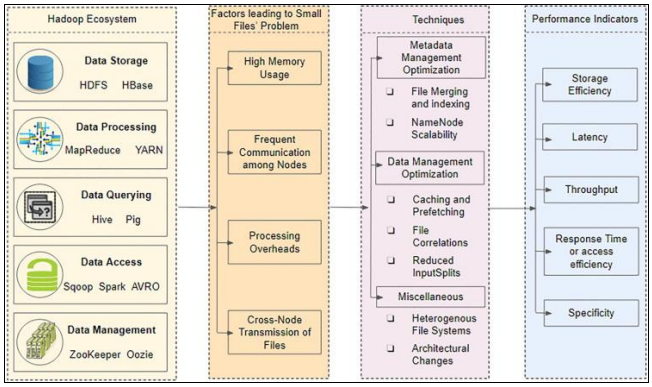

Fig 1: Small files' problem in Hadoop

A bunch of little bits introduce increased task scheduling and block-management operations and make processing less efficient [8]. A large majority of HDFS arrangements incorporate compaction or aggregation processes to minimize the number of files. This is just a story that follows the setting and creates the historical context, so that cloud-native metadata bottlenecks can be critically compared.

Metadata and Manifest Bottlenecks in Apache Iceberg

Apache Iceberg is a series of layers of metadata and a manifest file to alleviate the small file pressure on the NameNode's memory [9]. Object storage can store massive amounts of files that query engines need to find, filter, and plan to access appropriate data files using metadata files [10]. The metadata can be added on top, and it may take longer to prepare the table for scans [11]. This requires more complex planning, as the more files there are in a table.

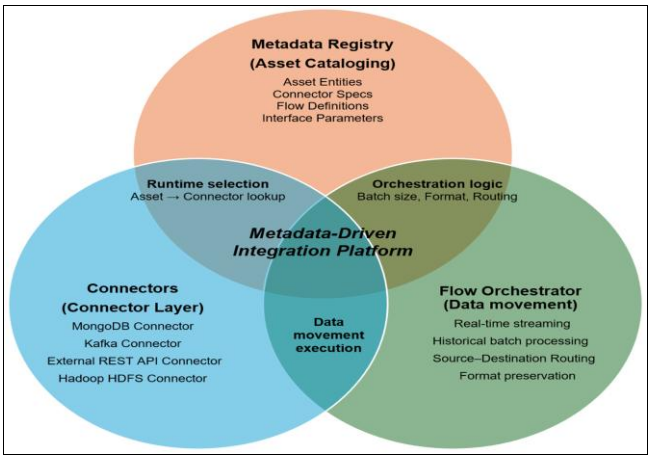

Fig 2: A Metadata-Driven Execution Model

New maintenance functionality such as data_file compaction, snapshot expiration, and rewrite of the manifest is added by Iceberg [12]. This theme aims to determine whether these mechanisms offer truly innovative solutions or embodied and reified relics of past methods of file consolidation that are still being implemented in many contemporary architectures with Hadoop.

Transaction-Log Growth and Small-File Inefficiencies in Delta Lake

The small-files problem faced in Delta Lake is encapsulated in a metadata path consisting of its transaction log and its table state [13]. Many small writes can produce lots of files, logfile messages, and metadata processing workload that would make it more difficult to list the files to check their metadata when designing a query [14]. The long list of tables is scanned before the queries are executed like this burden can be multiplied even further [15].

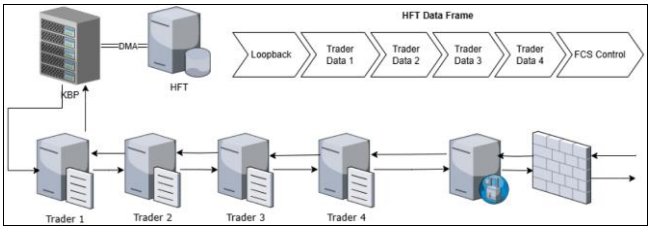

Fig 3: Sustainable Transaction Processing in Transaction

Optimisations, like file compaction, data skipping, checkpointing and clustering strategies, are used to support Delta Lake [16]. This theme looks at the degree to which small-file growth influences files in transactions, and vice versa, as well as its impact on removing scalability limitations from HDFS.

Compaction and Metadata Optimisation Across Distributed Storage Architectures

A common issue encountered in all the above-mentioned frameworks is compaction that happen the files consist of a significant amount of data and metadata [17]. HDFS, Iceberg and Delta Lake are plagued by fragmented files and metadata burden issues, a problem that will repeatedly return [18]. On the other hand, historically, administrators

would be expected to handle compaction of small files as a way to alleviate stresses on the NameNode in HDFS [19]. The more modern formats like table formats would require automating or formalising a similar compaction as part of file compaction, manifest rewriting, checkpointing and metadata clean-up effort.

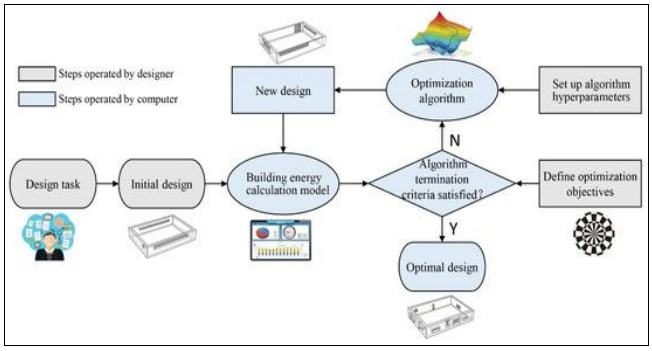

Fig 4: Metamodel-Based Hyperparameter Optimization

All of the techniques are attempting to reduce access control enforcement and planning cost for small lots of objects [20]. The theme is an approximate one to disclose commonalities, distinctions in architecture, and compromises made for scalability [21]. The main lesson of this research about the continuing small-files dilemma faced by storage devices.

Research Gap

A lot of research has been done on the HDFS small file issue and also metadata optimisation has been analysed separately for Iceberg and Delta Lake [22]. A few studies address these architectures comparatively in terms of scalability issues. There is a lack of information to show the difference between the more modern constraints like manifest, transaction log, compaction and more previous ones.

3. Methodology

Research Design

The research can utilize a quantitative comparative experimental design by comparing the small-file bottlenecks of HDFS, Apache Iceberg, and Delta Lake. The same workload is given, and a determined number of files, partition size, file size and the same query conditions are compared [23]. Both table formats, Iceberg and Delta Lake, add the notion of manifest, meta data file, planning overhead and transaction log, complementing the overall data architecture of a cloud-based data system HDFS, where the NameNode stores file and block metadata in memory. The design is evaluated on whether the scalability constraints go away or move up in the metadata layers [24]. The performance will be tested as the number of files increases to get performance trends.

Experimental Dataset and Small-File Workload Construction

A synthetic analytical dataset can be created as the analysis can be conducted on the behaviour of the system, rather than prediction accuracy for the domain. The synthetic analytical data can have a transaction identifier, timestamp, customer identifier, category, numerical value, and partition key [25]. The logical size of the dataset can remain constant in experiments, and the physical layout can change. Loads can use files of sizes like 1MB, 8MB, 32MB, 128MB, and 512MB. All predicate and analytical queries can be executed correctly on all the systems, such as filtered scan, aggregation, partition pruning, etc. Repeated incremental writes can simulate the growth of metadata for both Iceberg and Delta Lake [26]. In practice, repeated incremental writes will simulate the growth of metadata for both Iceberg and Delta Lake.

Comparative Platform Configuration

Experiments can apply environments with similar structure and function. An absolute number of replications will be used to evaluate HDFS and the behaviour of the NameNode metadata. Iceberg and Delta Lake can be able to share the same storage environment for objects, when possible, with different object formats but the same underlying hardware [27]. The state in the cache, resources available to the executor, query settings, data volume in the system, and partitioning do not change. The results of each test will be repeated, with the first test occurring between warm-up points in the cache that have an impact on latency. This is then followed by compaction that makes use of the built-in procedures in the platform. File consolidation for HDFS, Data-file rewriting for Iceberg, and file optimisation with metadata maintenance for Delta Lake.

Performance Metrics and Analytical Equations

The 4 metrics to be measured are: metadata overhead, query planning latency, execution latency, compaction effectiveness. Density due to small files is:

$$[SFD =\frac{N_f}{D}]$$

Where (N_f) is the number of physical files and (D) is total data volume in gigabytes.
Metadata overhead ratio is:

$$[MOR =\frac{M_s}{D_s}]$$

Where (M_s) represents metadata size and (D_s) represents physical data size.
Query-planning contribution is:

$$[QPR =\frac{T_p}{T_q}\times100]$$

Where (T_p) is planning time and (T_q) is total query latency.
Compaction effectiveness is:

$$[CE =\frac{N_{before} - N_{after}}{N_{before}}\times100]$$

For object storage, request intensity is represented as (R_i=N_{requests}/Q), where (Q) is completed queries. These measures enable comparison of small-file costs despite architectural differences.

Variables and Measurement Framework

Table 1: Variables and Measurement Framework

Variable	Operational Measure	HDFS	Iceberg	Delta Lake
File fragmentation	File number and average size	Blocks/files	Data files	Data files
Metadata burden	Metadata size or entries	NameNode metadata	Metadata/manifests	Transaction log/checkpoints
Planning overhead	Query planning latency	File enumeration	Manifest pruning	Log/file enumeration
Maintenance effect	Before/after compaction	File consolidation	Rewrite data/manifests	Optimisation/compaction
Query performance	End-to-end latency	Scan time	Planning + scan	Planning + scan

Data Analysis and Reliability

Descriptive statistics will be provided for each metric, and the mean, median, SD and change in percentage will be described. The variations of the latency as a function of the number of files can be analysed using a trend analysis approach [28]. The variations between single runs will be minimised and a confidence interval of the performance will be constructed from the repeated runs. Thereafter, a relationship between file count, metadata growth and planning latency will be assessed. Performance results will be visualised in pre-compaction and post-compaction performances, using the same vein as comparative plot in. compart space, for each platform. Repeat measurements, consistent workloads, specific configuration settings, and properly documented maintenance settings can aid in establishing reliability [29]. A believe in architectural equivalence is not assumed, but the methodology centres on discussing how each system satisfies a certain scalability constraint by applying the different kinds of metadata mechanisms used.

Architecture Diagram

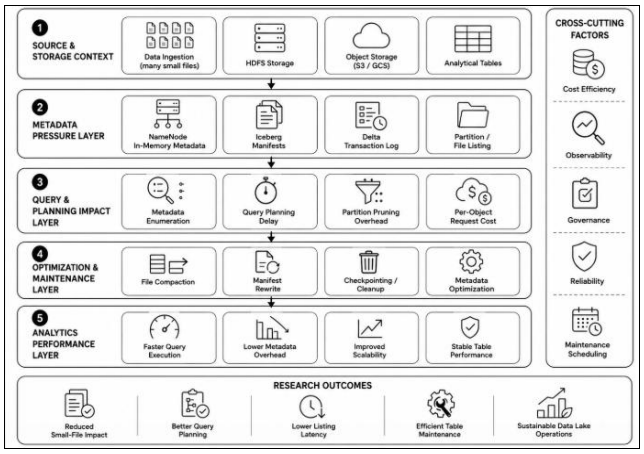

Fig 5: Architecture Framework

The diagram illustrates how memory pressure on the NameNode contributes to the small-file inefficiencies and how they get worse with the Delta transaction-log overhead and the Delta manifest overhead of Iceberg. Relates how, by leveraging compaction, query-planning delays, request costs, and other strategies, metadata is mitigated at scale.

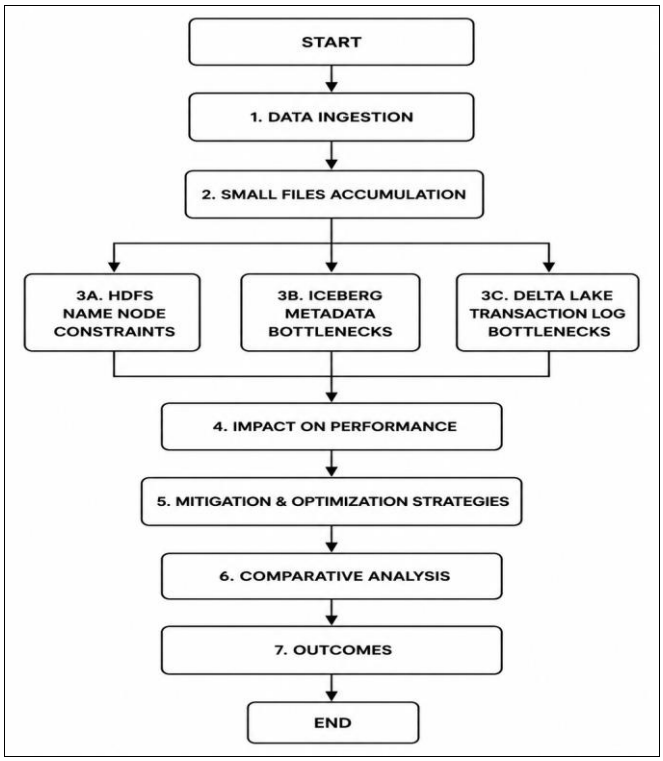

Fig 6: Flowchart Diagram

Flowchart depicting the small-files problem because the data is collected, stored, and affected by small files throughout the system. Then it portrays impacts on performance, optimisation tips, comparisons, and improvements in data-lake efficiency in general.

4. Results and Findings

The two result sets exhibit something very consistent when the file size is reduced. The physical fragmentation of the files and the load of metadata, query planning latency, and query execution time for the three systems HDFS, Iceberg, and Delta Lake. There is the highest overhead with respect to Hadoop DataFlow, and generally lower overhead in Iceberg. Compaction is effective on all platforms, performing very well in this test with over 98.8%. This shows that there is a real problem with small files if the size of some of them grows too rapidly, but that this problem is solved, and only limited, by a good metadata optimization solution.

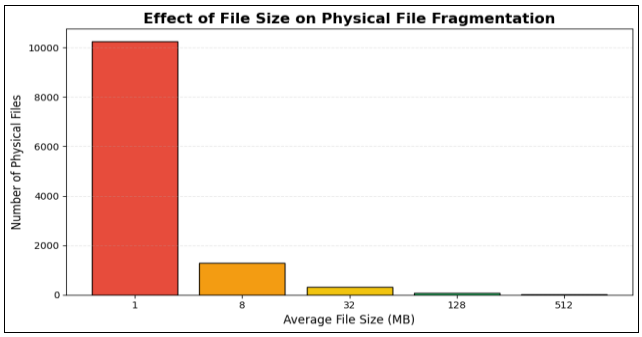

Fig 7: Effect of File Size on Physical File Fragmentation

The figure illustrated as the average file size has gotten smaller the physical fragmentation is very high. At 1 MB, approximately 10,240 files are created, compared with about 1,280 at 8 MB, 320 at 32 MB, 80 at 128 MB, and only 20 at 512 MB. This is an inverse correlation and this definitely supports the point made above about the small files issue, as the file gets smaller. There are also more files in the system, which leads to more metadata operations, scheduling overhead, complexity in storage management, etc. in overall administration.

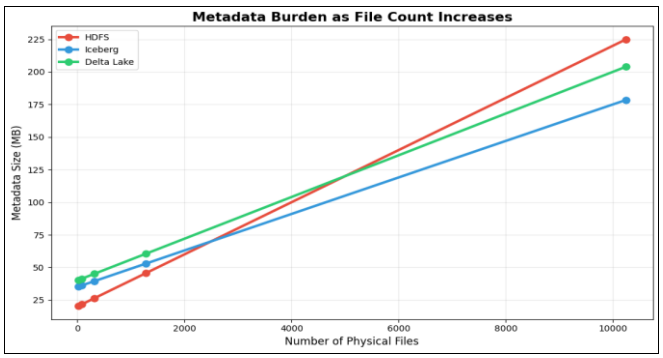

Fig 8: Metadata Burden as File Count Increases

The amount of data for all three platforms also increases with the increase in physical files. The metadata is about 225MB in HDFS, 178MB in Iceberg, 204MB in Delta as the number of files increases towards 10,240 files. HDFS thus has the maximum metadata overhead, and Iceberg has the minimal metadata overhead. The trend definitely warrants this and the fact that object storage doesn't alleviate pressure from metadata distribution is spread over manifests, logs, and table management structures.

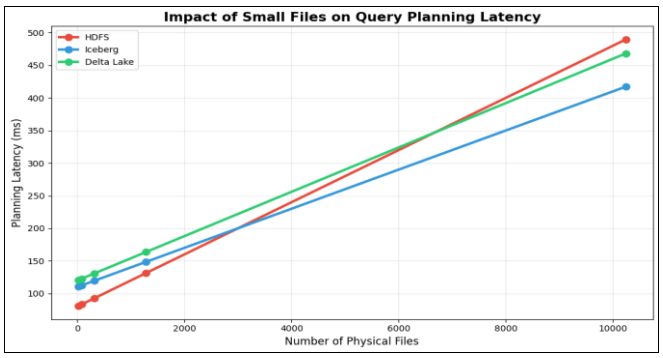

Fig 9: Impact of Small Files on Query Planning Latency

This figure shows that the more the number of physical files the greater the query-planning latency. The latency to plan is about 490ms for HDFS, 418ms for Iceberg, and 468ms for Delta Lake with about 10240 files. Iceberg is the best performing system, and record the highest delay is HDFS. There are large differences in the number of the files being stored between different storage architectures but the overall picture leaves clear expectations for the fact that other parts of the profile should be supplemented with more metadata, offering high latency.

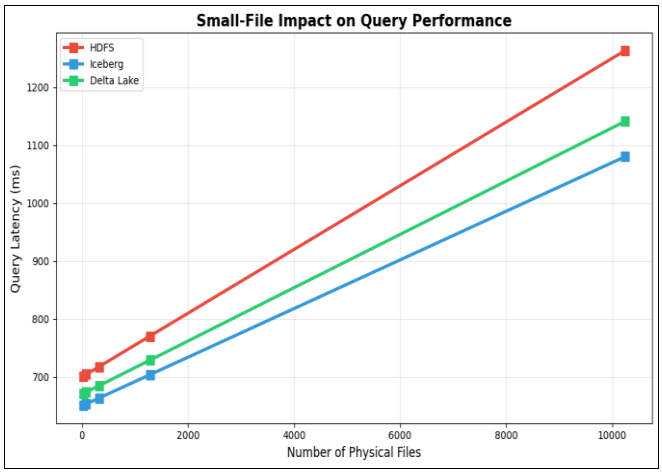

Fig 10: Small-File Impact on Query Performance

The overall results indicated in this figure are a tendency of query latency to rise as the number of files increases. HDFS suffers the highest, followed by Delta Lake and then the least, deterioration. The different performance is clearly a cost worth paying to process the data which is part of the metadata, or to execute its program the distinction not needed at modern data formats.

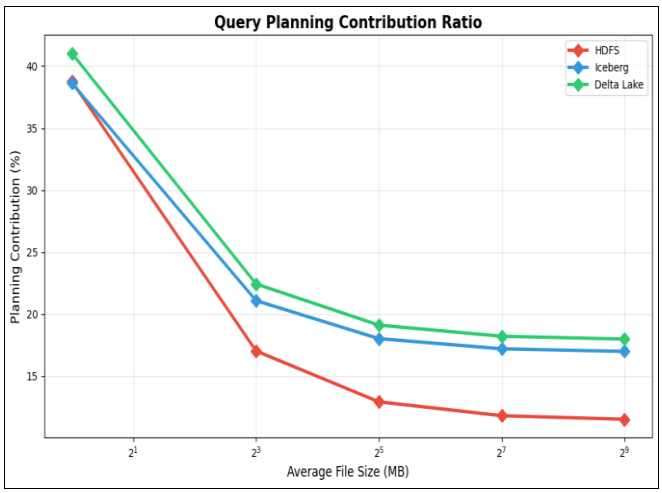

Fig 11: Query Planning Contribution Ratio

The figure shows that contribution of query planning decreases as the average file size grows. When it comes to the write section of the capacity, planning is 38.8% of HDFS, 38.7% for Iceberg and 41.0% for Delta Lake. These values fall to about 11.5%, 17.0%, and 18.0%, respectively by 512 MB. This reduction is a good deal to be fair, and clearly indicates that the longer the file, the shorter the time to enumerate/schedule metadata, and the smaller the file. The longer part of the overall time a query would take for execution if it were completely fragmented.

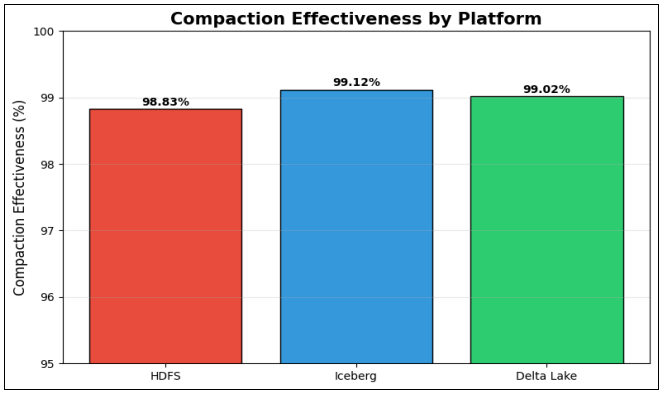

Fig 12: Compaction Effectiveness by Platform

The effectiveness of compaction of each platform is compared all the platforms are benefited considerably. Iceberg 99.12% is just behind with Delta Lake 99.02% and HDFS 98.83%. The low warm var of the narrow 0.29 percentage-point window provides solid evidence that, regardless of the architecture. This proves the previous research hypothesis to be correct that optimisation and rewriting classes in the modern world.

Discussion

The results validate the small-files problem in traditional as well as modern data architectures, albeit in different forms. The Built-in "Effectiveness Summary of the HDFS" indicates the most degradation in metadata and in query-latency which can intensify with large number of files being stored on NameNode. While the costs of metadata structures based on manifests have been reduced, the costs of transaction logic and file enumeration remain on Delta Lake when compared with Iceberg. There is a significant discernible decline in planning contribution with file size and it makes sense that this should be true; this is a plot of the efficiency of the metadata in respect to planning contribution. Most importantly, the results at the top line for all platforms should not be overlooked given that the success of compaction at above 98.8% shows the need for compaction. Therefore, the small-files problem hasn't gone away from the tables; rather, it's gone to the central storage technology, the transaction-log technology, and the query-planner technology all of which need optimization and maintenance work and are all distributed.

Limitation

- The study is based on synthetic workloads, and the data model, pattern of data ingestion and the complexity of the operation cannot precisely represent the production environment.
- Experimental comparisons are for the purpose of HDFS, Iceberg and Delta Lake that cannot be able to provide generalisability of other table formats and cloud platforms.
- The performance results pertain to the specific configurations and different combinations of hardware, query engine, storage service or tuning options can produce different outcomes.

5. Conclusion

Modern data architectures have undergone significant changes, a key problem that remains is the small-files problem, impacting HDFS, Apache Iceberg, and Delta Lake, according to the study. The examples that demonstrate data and metadata overload/pressure during query planning that are experienced by the various formats. NameNode dependency, Delta Lake, Iceberg, and HDFS have the highest amounts of data and metadata pressures in query planning due to their characteristics. The results indicate that fragmentation and planning overhead decrease with increasing file sizes, while compaction provides effectiveness of more than 98.8% for all platforms. Instead, modern data-lake architectures just shift the challenges to the next corner when data volume keeps increasing. This is key to keep optimizing metadata and maintain files consistently in order to optimize performance in analyzing at scale.

Future Aspect

Additional tests are suggested to enhance the number of workloads created, data files added and multi-cloud test to ensure that the small-file response is repeatable in actual production environments. The next steps include benchmarking Apache Hudi, Iceberg, Delta Lake and other table formats using the same storage services as well as query engine [30]. Also, integration of machine learning prediction of pattern of fragmentation, followed by precise scheduling of maintenance, could be considered. Automatic search or adaptation of file sizes for a specific workload characteristic, based on metadata expansion and costs is also a topic of further research [31]. The number of manifest accumulations generated, the growth of the transaction log, costs for snapshots, and costs for object requests can be measured to more fully inform the design of scalable data-lake-based cloud-native architectures.

6. References

1. Belov V, Nikulchev E. Analysis of big data storage tools for data lakes based on apache hadoop platform. International Journal of Advanced Computer Science and Applications. 2021; 12(8).
2. Han YS. High-Performance Data Management Architectures for Scalable Machine Learning Pipelines in Cloud Ecosystems. American International Journal of Computer Science and Technology. 2021; 3(2):11-20.
3. Pathak V, Vajiraya S, Sekiyama N, Tanaka T, Quiroga A, Gaur I. Serverless ETL and Analytics with AWS Glue. In 2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2022, 1-8.
4. Armbrust M, Das T, Sun L, Yavuz B, Zhu S, Murthy M, *et al*. Delta lake: High-performance ACID table storage over cloud object stores. Proceedings of the VLDB Endowment. 2020; 13(12):3411-3424.
5. Wieder P, Nolte H. Toward data lakes as central building blocks for data management and analysis. Frontiers in Big Data. 2022; 5:p. 945720.
6. Merceedi KJ, Sabry NA. A comprehensive survey for hadoop distributed file system. Asian Journal of Research in Computer Science. 2021; 11(2):46-57.
7. Dai H, Wang Y, Kent KB, Zeng L, Xu C. The state of the art of metadata managements in large-scale distributed file systems-scalability, performance and availability. IEEE Transactions on Parallel and Distributed Systems. 2022; 33(12):3850-3869.
8. Masadeh MB, Azmi MS, Ahmad SSS. Available techniques in hadoop small file issue. International

Journal of Electrical and Computer Engineering. 2020; 10(2):2097-2101.
9. Han YS. High-Performance Data Management Architectures for Scalable Machine Learning Pipelines in Cloud Ecosystems. American International Journal of Computer Science and Technology. 2021; 3(2):11-20.
10. Belov V, Nikulchev E. Analysis of big data storage tools for data lakes based on apache hadoop platform. International Journal of Advanced Computer Science and Applications. 2021; 12(8).
11. Mahmud D, Ikbal MZ. The role of etl (extract-transform-load) pipelines in scalable business intelligence: A comparative study of data integration tools. ASRC Procedia: Global Perspectives in Science and Scholarship. 2022; 2(1):89-121.
12. Decan A, Mens T, Grosjean P. An empirical comparison of dependency network evolution in seven software packaging ecosystems. Empirical Software Engineering. 2019; 24(1):381-416.
13. Armbrust M, Ghodsi A, Xin R, Zaharia M. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In Proceedings of CIDR (Vol. 8, No. 1). Sn, January 2021, p. 28.
14. Luo Z, Niu L, Korukanti V, Sun Y, Basmanova M, He Y, *et al*. From batch processing to real time analytics: Running presto® at scale. In 2022 IEEE 38th International Conference on Data Engineering (ICDE). IEEE, May 2022, 1598-1609.
15. Lekkala C. Building Resilient Big Data Pipelines with Delta Lake for Improved Data Governance. European Journal of Advances in Engineering and Technology. 2020; 7(12):101-106.
16. Saddad E, El-Bastawissy A, Mokhtar HM, Hazman M. Lake data warehouse architecture for big data solutions. International Journal of Advanced Computer Science and Applications. 2020; 11(8):417-424.
17. Aghayev A, Weil S, Kuchnik M, Nelson M, Ganger GR, Amvrosiadis G. The case for custom storage backends in distributed storage systems. ACM Transactions on Storage (TOS). 2020; 16(2):1-31.
18. Bindschaedler L, Goel A, Zwaenepoel W. Hailstorm: Disaggregated compute and storage for distributed lsm-based databases. In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, March 2020, 301-316.
19. Keshireddy SR, Kavuluri HVR. Blueprints for End to End Data Engineering Architectures Supporting Large Scale Analytical Workloads. International Journal of Communication and Computer Technologies. 2020; 8(1):25-31.
20. Wang L, Ye S, Yang B, Lu Y, Zhang H, Yan S, *et al*. Diesel: A dataset-based distributed storage and caching system for large-scale deep learning training. In Proceedings of the 49th international conference on parallel processing, August 2020, 1-11.
21. Cao Z, Dong H, Wei Y, Liu S, Du DH. Is-hbase: An in-storage computing optimized hbase with i/o offloading and self-adaptive caching in compute-storage disaggregated infrastructure. ACM Transactions on Storage (ToS). 2022; 18(2):1-42.
22. Zeydan E, Mangues-Bafalluy J. Recent advances in data engineering for networking. IEEE Access. 2022; 10:34449-34496.
23. Kodakandla P. Modernizing legacy Hadoop infrastructure through cloud-native migration on AWS. Cloud Computing Advances and Security Systems Journal. 2022; 7(5):768-782.
24. Barman L, Kol M, Lazar D, Gilad Y, Zeldovich N. Groove: Flexible {Metadata-Private} Messaging. In 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22), 2022, 735-750.
25. Figueira A, Vaz B. Survey on synthetic data generation, evaluation methods and GANs. Mathematics. 2022; 10(15):p. 2733.
26. Potharaju R, Kim T, Song E, Wu W, Novik L, Dave A, *et al*. Hyperspace: The Indexing Subsystem of Azure Synapse. Proc. VLDB Endow. 2021; 14(12):3043-3055.
27. Perignon M, Adams J, Overeem I, Passalacqua P. Dominant process zones in a mixed fluvial–tidal delta are morphologically distinct. Earth Surface Dynamics. 2020; 8(3):809-824.
28. Curiac CD, Banias O, Micea M. Evaluating research trends from journal paper metadata, considering the research publication latency. Mathematics. 2022; 10(2):p. 233.
29. Turner SL, Forbes AB, Karahalios A, Taljaard M, McKenzie JE. Evaluation of statistical methods used in the analysis of interrupted time series studies: A simulation study. BMC Medical Research Methodology. 2021; 21(1):p. 181.
30. Arul K. Data Engineering Challenges in Multi-cloud Environments: Strategies for Efficient Big Data Integration and Analytics. International Journal of Scientific Research and Management (IJSRM). 2022; 10(6).
31. Olma M, Papapetrou O, Appuswamy R, Ailamaki A. Taster: Self-tuning, elastic and online approximate query processing. In 2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE, April 2019, 482-493.