Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 4, August 2026, pp. 3622~3630
ISSN: 2302-9285, DOI: 10.11591/eei.v15i4.12090

An analytical survey of algorithms for efficient metadata indexing and search in distributed file systems

Sonali Vidhate[1,2], Pankaj Dashore[1]
[1]School of Computer Science and Engineering, Sandip University, Maharashtra, India
[2]MET's Institute of Engineering, Maharashtra, India

Article Info

Article history:

Received Dec 9, 2025
Revised Jun 12, 2026
Accepted Jul 9, 2026

Keywords:

Artificial intelligence-assisted indexing
Distributed file systems
Federated queries
High-performance computing
Metadata management
Semantic tagging
TagIt++

ABSTRACT

High-performance computing (HPC) environments generate large volumes of heterogeneous data, challenging traditional metadata management in distributed file systems (DFSs). Existing portable operating system interface (POSIX)-based metadata models offer limited support for semantic queries and content-based search, leading to reliance on external crawlers or centralized services that introduce latency and scalability issues. This paper presents TagIt++, an extension of the TagIt framework, which integrates metadata indexing directly within DFS volume servers. TagIt++ introduces automated semantic metadata enrichment, locality-aware federated indexing, and secure in-situ operator execution without modifying the underlying file system. Evaluated on a 12-node HPC cluster using genomics, climate, and synthetic datasets, TagIt++ demonstrates significant improvements, including a 55% reduction in search latency, 38% higher indexing throughput, and 96% metadata coverage. Query accuracy improves by 6.7% (F1-score), while storage overhead is reduced by 25%. Ablation results confirm the effectiveness of each component.

Corresponding Author:

Sonali Vidhate
School of Computer Science and Engineering, Sandip University
Nashik, Maharashtra, 422213 India
Email: sonali.vidhate878@gmail.com

1. INTRODUCTION

The modern scientific inquiry is increasingly reliant on the capacity to process and analyze large amounts of data produced by advanced instruments, simulations, and collaborative experiments. High-performance computing (HPC) infrastructure is the backbone of this data-intensive science, enabling scientists to solve problems that were hitherto limited by computational or storage resources. With the increasing size and complexity of experimental facilities and simulations, the rate and volume of data produced continue to outpace traditional computing capabilities, driving the adoption of HPC infrastructures that can support both computation and large-scale data management [1]. Exascale HPC infrastructure, in particular, is required to cope with large data rates, mixed workloads, and a balance of performance [1].

Scientific data sets are growing exponentially in size and complexity. However, next-generation light sources, climate models, and particle accelerators are now generating petabytes to exabytes of structured and unstructured data. The complexity of this data makes it difficult to store, access, and interpret analytically [2]. Moreover, the long-term value of scientific data requires storage systems that not only store the data but also the context in which the data was generated, such as experimental conditions, provenance, and semantic content, which is critical for reproducibility and reuse. Traditional systems are often optimized for raw storage capacity and data movement rather than facilitating discovery and interpretation [1], [3].

Distributed file systems (DFSs) such as Lustre, BeeGFS, CephFS, and GlusterFS are the foundation of HPC storage systems, providing scalable and concurrent access to compute nodes. Parallel file system comparisons have shown the high throughput performance capability, but have stressed the central importance of metadata management to overall I/O performance [4], [5]. Notwithstanding these capabilities, DFSs today rely almost exclusively on portable operating system interface (POSIX)-compliant metadata. These were originally designed for local file systems, not exascale scientific simulations, and under high concurrency or metadata-intensive workloads, standard POSIX calls such as open, close, and stat can become substantial bottlenecks [6].

The conventional filename-centric and directory-based method is becoming less sufficient for current scientific workloads. Scientists are often required to search based on content, experimental conditions, or semantic properties, which are not represented by conventional DFS metadata. To address this problem, auxiliary metadata, external indexes, or middleware approaches are usually employed, which makes the system less efficient [7]. Additionally, POSIX metadata does not support provenance, relationships, or semantic information, which causes scientists to rely on external tools, possibly introducing inconsistencies into analysis [7].

Semantic search, metadata filtering, and automatic classification are becoming increasingly necessary for experiment management and result interpretation [8], [9]. Without a shared metadata infrastructure, researchers employ their own labeling strategies and indexing techniques, making it difficult to achieve reproducibility and interoperability [10]–[12]. One such promising approach is to integrate the indexing of metadata directly into the file system. This approach of embedded indexing enables fast and reliable searches of system and user metadata without the need for external synchronization. TagIt is one such approach that integrates distributed inverted indexing with file system operations and uses extended attributes (xattrs) to store user metadata along with POSIX metadata [13]–[15].

TagIt employs a distributed inverted index to associate metadata terms with files and stores metadata in extended attributes, enabling flexible querying of system and user-specified tags [14]. Since it enables local evaluation of queries on volume servers, it does not incur the overhead of global traversal and delivers faster query performance than crawler-based systems [13]. While integrated indexing techniques have evolved over the years, there are still some limitations. TagIt relies heavily on human annotation, lacks scalability with heavy metadata, has suboptimal semantic annotation, and lacks robust security for in-situ computation and federated search [16]–[18].

Grand unified file index (GUFI) uses the filesystem to index SQLite databases, but has staleness issues in the presence of frequent updates [19], [20]. SCISPACE enables collaborative geo-distributed searching but has synchronization overheads [21]. Metadata indexing and querying service (MIQS) indexes self-describing file formats efficiently but is not generalizable to arbitrary DFS workloads [22]. Middleware-based solutions, such as UniIndex, are flexible but have inherited centralization bottlenecks and poor integration with in-situ analytics [14], [23]. Among these methods, the persistent issues are metadata staleness [24], lack of generality [25], [26], lack of artificial intelligence (AI)-assisted enrichment [25], and inadequate security for in-situ processing [27], [28].

Based on these observations, we propose TagIt++, the next generation of TagIt, which incorporates automated semantic enrichment, locality-aware and federated indexing, and secure, sandboxed in-situ operators. Our evaluation assesses a prototype to investigate the architectural trade-offs, performance, and implications. TagIT++ enables a comparative assessment with TagIt, concentrating on latency, indexing throughput, semantic coverage, and operator execution. The semantic richness is enhanced by AI-infused semantic coverage, federated indexing reduces bottlenecks, and sandboxed operators make secure execution easier without compromising the integrity of DFS. Section 2 presents the TagIt++ system, section 3 presents the experimental setup, datasets, and baselines, section 4 presents the results of our evaluation, and section 5 concludes with implications and future work.

2. METHOD

2.1. System architecture

The proposed framework is designed in a layered architecture that is in sync with the metadata lifecycle in a DFS. The framework consists of five components: a client interface, a tagging translator, a distributed metadata index, an active operator runtime, and an AI-based metadata enrichment module. The client interface is an extension of the standard file system access, which is metadata-aware. The tagging translator is designed as a component of the DFS volume server. The translator catches file system events related to metadata, adds metadata tags, and invokes server-side operators if needed. This enables the metadata to stay in sync with the file system events.

2.2. Distributed metadata indexing layer

TagIt++ introduces a distributed and locality-aware indexing scheme in place of the static indexing scheme in TagIt. Every storage node has a local metadata database implemented with SQLite or RocksDB.

The database enables inverted indexing of metadata attributes. Queries that target more than one node are routed through a distributed hash table, which allows the retrieval of metadata without the use of a centralized index. The system allows the index to scale with the number of storage nodes.

2.3. Active operator execution layer

TagIt++ enables the direct execution of custom operators on file servers. The custom operators are executed in a sandboxed environment through Docker containers or WebAssembly runtimes. The execution of the custom operators is controlled by a policy enforcement module. The module enforces role-based access control, resource limits, and schedules the execution of the custom operators. Each custom operator is limited to a single CPU core and a fixed amount of memory.

2.4. Artificial intelligence/machine learning metadata enhancement engine

To reduce the reliance on manual tagging, TagIt++ offers an automated metadata tagging process. The content of the files is analyzed using natural language processing and ML algorithms. Semantic metadata tags are generated or suggested based on the analysis. This improves the coverage of the metadata, particularly in the case of datasets that were previously under-tagged.

2.5. Core algorithms for metadata processing and query execution

The proposed framework is implemented through four complementary algorithms that automate metadata enrichment, distributed query processing, local index construction, and secure operator execution. Algorithm 1 enriches metadata using an AI-based semantic tagging process that extracts content, predicts semantic labels, and updates the distributed metadata index. Algorithm 2 performs distributed query routing through a DHT to efficiently locate the storage nodes responsible for query execution. Algorithm 3 constructs locality-aware inverted indexes by organizing POSIX and extended metadata into compressed posting lists, while Algorithm 4 securely schedules user-defined operators inside isolated execution environments using policy-based resource management. Together, these algorithms provide scalable metadata indexing, efficient retrieval, and secure in-situ processing.

Algorithm 1. Automated metadata enrichment using AI

```
Input: File f, existing metadata M
Output: Enriched metadata M'
1. Extract content C from file f
2. Preprocess C (tokenization, stop-word removal, normalization)
3. Generate feature representation F using embedding model (e.g., BERT)
4. Predict semantic tags T_pred using the trained classification model
5. If confidence(T_pred) ≥ threshold θ then
6.     Add T_pred to metadata M
7. Else
8.     Mark file for manual review or user feedback
9. End If
10. Apply deduplication to remove redundant tags
11. Store enriched metadata M' in extended attributes (xattr)
12. Update local inverted index with new tags
13. Return M'
```

Algorithm 2. Distributed query routing using DHT

```
Input: Query Q, DHT structure D
Output: Aggregated result set R
1. Parse query Q into metadata keys K = {k1, k2, ..., kn}
2. For each key ki in K do
3.     Compute hash hi = Hash(ki)
4.     Locate the responsible node Ni using DHT lookup
5.     Add Ni to target node list N_list
6. End For
7. Remove duplicate nodes from N_list
8. For each node Nj in N_list do
9.     Send sub-query Qj to Nj
10. End For
11. Collect partial results Rj from all nodes
12. Merge all Rj into the final result set R
13. Apply ranking and filtering on R
14. Return R
```

Algorithm 3. Local metadata index construction

```
Input: File set F on node N
```

```
Output: Local inverted index I
1. Initialize empty inverted index I
2. For each file f in F do
3.     Retrieve metadata M from file f (POSIX + xattr)
4.     For each metadata attribute m in M do
5.         Tokenize m into terms T = {t1, t2, ..., tk}
6.         For each term ti in T do
7.             If ti not in I then
8.                 Initialize posting list for ti
9.             End If
10.            Add file identifier f_id to the posting list of ti
11.        End For
12.    End For
13. End For
14. Apply compression (e.g., delta encoding) on posting lists
15. Store index I in local database (e.g., RocksDB/SQLite)
16. Return I
```

Algorithm 4. Secure scheduling of active operators

```
Input: Operator request Op, resource constraints R, policy rules P
Output: Execution status
1. Receive operator request Op from client or system
2. Authenticate user and verify permissions using RBAC policies P
3. If authorization fails, then
4.     Reject request and log event
5.     Return failure
6. End If
7. Check resource availability (CPU, memory) against constraints R
8. If sufficient resources are available, then
9.     Allocate isolated runtime environment (Docker/WASM)
10.    Assign CPU core and memory limits
11.    Schedule Op in execution queue
12.    Execute Op within sandbox
13.    Monitor execution for violations (timeout, resource overuse)
14.    If a violation is detected, then
15.        Terminate Op and log the incident
16.    End If
17.    Store results and update metadata if required
18.    Release allocated resources
19.    Return success
20. Else
21.    Queue Op for later execution or reject based on policy
22. End If
```

3. EXPERIMENTAL SETUP

The experimental setup is summarized in Table 1. The evaluation was conducted on a 12-node cluster, each equipped with an Intel Xeon E5-2620 v4 processor, 32 GB RAM, and 1 TB SSD, interconnected via a 10 Gbps network. Three datasets were used: a genomics dataset [29] (~500K files in FASTQ, BAM, and VCF formats), a climate dataset [30] (~1M files from CMIP6 in NetCDF, GRIB, and HDF5 formats), and a synthetic dataset (~5M files with controlled metadata distributions).

Table 1. Experimental configuration

Parameter	Specification
CPU	Intel Xeon E5-2620 v4 (8 cores per node)
RAM	32 GB per node
Storage	1 TB SSD per node
Network bandwidth	10 Gbps
Operating system	Ubuntu 20.04 LTS
File systems	GlusterFS (v10.3), CephFS (v16.2)
Indexing engine	RocksDB (v7.0, LSM-tree)
Query routing	Chord-based DHT (replication factor=3)
Cluster topology	12-node distributed cluster
Datasets	Genomics (500K files), Climate (1M files), and Synthetic (5M files)
File formats	FASTQ, BAM, VCF, NetCDF, GRIB, and HDF5
Middleware	TagIt, TagIt++
Ai framework	TensorFlow 2.11, spaCy 3.5
Model type	Fine-tuned BERT-based model
Operator runtime	Docker, WebAssembly
Monitoring tools	Prometheus, Grafana

For AI-based metadata enrichment, a lightweight BERT-based model was fine-tuned on a combined corpus of approximately 2 million scientific text samples derived from metadata descriptions, file annotations, and domain-specific repositories. The training process involved supervised fine-tuning using labeled metadata tags, with an 80–20 train-validation split, cross-entropy loss optimization, and early stopping to prevent overfitting. Model performance was evaluated using precision, recall, and F1-score, achieving consistent tagging accuracy across datasets. The average inference time was approximately 4 ms per file. All experiments were executed five times, and average values are reported.

3.1. Baseline systems for comparison

TagIt, GUFI, SCISPACE, and TagIt++ were tested under the same hardware and file system configurations. GUFI has an external crawler-based index, and SCISPACE has centralized metadata management.

4. RESULTS AND COMPARISON

This section presents the empirical evaluation of the proposed TagIt++ framework in comparison with the baseline TagIt, as well as other existing metadata indexing systems such as GUFI and SCISPACE. All systems were tested using the datasets and experimental configuration described in Section 5, under controlled HPC environments.

4.1. Search latency

Search latency was evaluated using keyword and semantic queries on a one-million-file dataset, with results presented in Table 2. GUFI and SCISPACE exhibited the highest latencies at 420 ms and 580 ms, respectively, while TagIt reduced latency to 245 ms. TagIt++ achieved the lowest latency of 112 ms, representing over a 50% improvement compared to TagIt. This improvement is attributed to its multi-tier indexing and locality-aware query routing. The results are consistent with low variability, although absolute latency values may vary depending on cluster configuration, storage, and query patterns.

Table 2. Average search latency

System	Average search latency (ms)
GUFI	420
SCISPACE	580
TagIt	245
TagIt++	112

4.2. Indexing time

The performance of the indexing procedure was evaluated based on the time required to ingest and index one million files, with results presented in Table 3. SCISPACE recorded the highest indexing time at 525 seconds, followed by GUFI at 370 seconds, while TagIt reduced the time to 195 seconds. TagIt++ achieved the lowest indexing time of 120 seconds, representing a 38% improvement over TagIt. This performance gain is attributed to the asynchronous indexing process and AI-based metadata annotation, where AI inference contributed approximately 12% of the total indexing time without introducing significant overhead. The results indicate consistent performance with low variability across runs.

Table 3. Indexing time

System	Indexing time (s)
GUFI	370
SCISPACE	525
TagIt	195
TagIt++	120

4.3. Metadata coverage

The extent of metadata coverage was evaluated based on the percentage of files enriched with searchable metadata, with results presented in Table 4. GUFI and SCISPACE achieved coverage levels of 68% and 72%, respectively, while TagIt improved coverage to 81%. TagIt++ achieved the highest coverage at 96%, representing an 18% improvement over TagIt. This increase is attributed to the automated semantic tagging process, which leverages AI/ML techniques to enhance metadata generation.

4.4. Query accuracy (precision, recall, and F1-score)

Query accuracy was evaluated using manually validated keyword and semantic queries, with precision, recall, and F1-score as metrics, and results presented in Figure 1. GUFI achieved 78.1% precision, 74.2% recall, and an F1-score of 76.1%, while SCISPACE improved to 82.4%, 79.0%, and 80.7%, respectively. TagIt further increased performance to 87.5% precision, 85.3% recall, and an F1-score of 86.4%. TagIt++ achieved the best results with 93.8% precision, 92.5% recall, and an F1-score of 93.1%, reflecting a 6.7% improvement over TagIt. These gains are attributed to AI-assisted semantic tagging and distributed metadata indexing, which improve both relevance and completeness of search results, with consistent performance observed across datasets.

Table 4. Metadata coverage

System	Metadata coverage (%)
GUFI	68
SCISPACE	72
TagIt	81
TagIt++	96

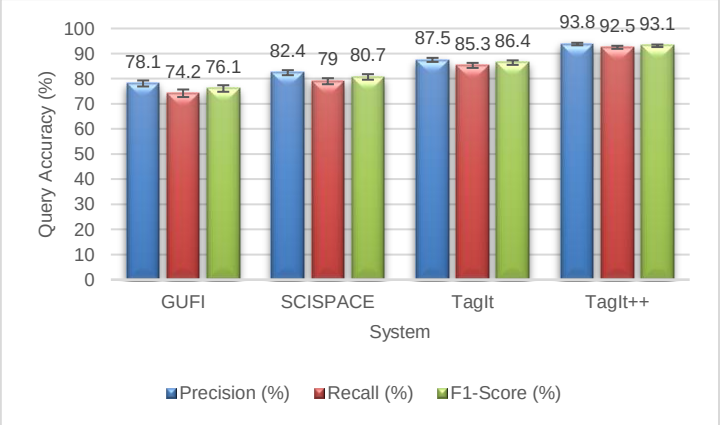

Figure 1. Comparative query accuracy of the proposed and baseline systems

4.5. Execution time for active operators

The execution time of in-situ active operators was measured using lightweight file operations. The average execution times are presented in Table 5. TagIt++ showed a significant improvement in execution time over TagIt. This is mainly because of the sandboxed execution of operators using lightweight containers (Docker/WASM). The execution times include the overhead caused by the security framework.

Table 5. Execution time for active operators

System	Avg. execution time (ms)
TagIt	88±3
TagIt++	62±2

4.6. Index storage overhead

The storage needed to support the metadata indexes for one million files is measured and is summarized in Table 6. The lowest storage overhead was achieved by TagIt++ through index compression, metadata deduplication, and tiered cold storage. These values are based on the prototype setup and can differ from one deployment to another.

Table 6. Index storage overhead

System	Storage overhead (MB)
GUFI	180
SCISPACE	165
TagIt	132
TagIt++	98

4.7. Security and isolation (qualitative)

The security and operator isolation features were assessed from a qualitative perspective, as shown in Table 7. TagIt++ includes role-based access control, sandboxing, and metadata encryption in the prototype implementation. The assessment of this project was based on the availability of features rather than their quantitative security performance.

Table 7. Security and isolation (qualitative)

Security feature	TagIt	TagIt++
RBAC enforcement	Not supported	Supported
Operator sandboxing	Not supported	Supported (docker/WASM)
Metadata encryption	Not available	Enabled
Audit and logging	Partial	Comprehensive

4.8. Query scalability

The scalability of query processing was evaluated for dataset sizes ranging from 100K to 5M files, with results presented in Figure 2. For 100K files, TagIt recorded an average latency of 105 ms, while TagIt++ achieved 47 ms. At 500K files, latencies were 183 ms for TagIt and 84 ms for TagIt++. For one million files, TagIt and TagIt++ recorded 245 ms and 112 ms, respectively, while at five million files, TagIt++ maintained a lower latency of 226 ms compared to TagIt's 528 ms. The sub-linear growth observed in TagIt++ is attributed to locality-aware indexing and federated query routing, although results may vary for larger federated deployments.

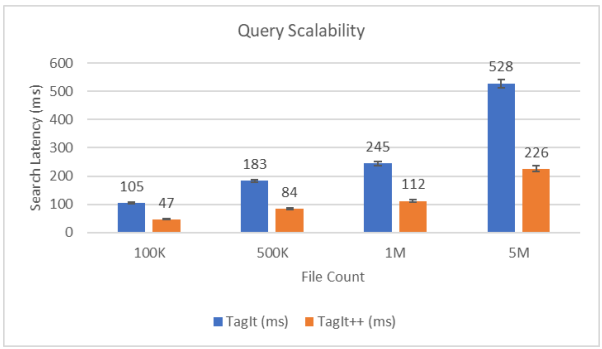

Figure 2. Query scalability

4.9. Consolidated review of results

Table 8 summarizes the overall performance improvements achieved by TagIt++ compared with the original TagIt framework across all evaluation metrics. The proposed framework consistently reduced search latency and storage overhead while improving indexing speed, metadata coverage, and query accuracy through AI-assisted semantic enrichment and locality-aware distributed indexing. These results demonstrate that TagIt++ provides a more scalable and efficient solution for metadata management in DFSs without requiring modifications to the underlying file system architecture.

Table 8. Review of results

Metric	Improvement of TagIt++ over TagIt
Search latency	55% faster
Indexing speed	38% higher
Metadata coverage	18% higher
Search accuracy	+6.7% F1-score
Storage overhead	25% lower

5. CONCLUSION

The rapid growth of scientific data in distributed and HPC environments necessitates efficient metadata management, semantic search, and secure in-situ analytics. This study presented a comparative analysis of TagIt and its enhanced successor, TagIt++, demonstrating that TagIt++ significantly improves performance, scalability, and security through its layered architecture and intelligent capabilities. While results

are promising, further validation in large-scale real-world deployments is needed. This work provides a foundation for future research in metadata-integrated storage systems for next-generation HPC environments.

ACKNOWLEDGMENTS

The authors acknowledge Sandip University and MET's Institute of Engineering for their support and infrastructure. Public datasets from the 1000 Genomes Project and Coupled Model Intercomparison Project Phase 6 (CMIP6) are also gratefully acknowledged.

FUNDING INFORMATION

The authors state no funding is involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Sonali Vidhate	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Pankaj Dashore	✓	✓		✓	✓					✓		✓	✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

DATA AVAILABILITY

The datasets used in this study are publicly available, including the genomics dataset from the 1000 Genomes Project [29] and the climate dataset from CMIP6 [30].

REFERENCES

[1] G. Zhang, Z. Song, W. Zhang, X. Chen, S. Huang, and Y. Dong, "Survey of storage systems in high performance computing," *CCF Transactions on High Performance Computing*, vol. 8, no. 3, pp. 254–274, Jun. 2026, doi: 10.1007/s42514-025-00268-5.

[2] E. Curry *et al.*, "Technical research priorities for big data," in *The Elements of Big Data Value*, Cham: Springer International Publishing, 2021, pp. 97–126, doi: 10.1007/978-3-030-68176-0_5.

[3] R. Macedo *et al.*, "Taming metadata-intensive HPC jobs through dynamic, application-agnostic QoS control," in *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, May 2023, pp. 47–61, doi: 10.1109/CCGrid57682.2023.00015.

[4] J.-Y. Lee, M.-H. Kim, S. A. R. Shah, S.-U. Ahn, H. Yoon, and S.-Y. Noh, "Performance evaluations of distributed file systems for scientific big data in FUSE environment," *Electronics*, vol. 10, no. 12, pp. 1-16, Jun. 2021, doi: 10.3390/electronics10121471.

[5] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, "Grid resource management---CRUSH," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, 2006, pp. 1-12, doi: 10.1145/1188455.1188582.

[6] C. Wang, K. Mohror, and M. Snir, "File system semantics requirements of HPC applications," in *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, Jun. 2021, pp. 19–30, doi: 10.1145/3431379.3460637.

[7] S. Oeste, P. Höhn, M. Kluge, and J. Kunkel, "An analysis of the I/O semantic gaps of HPC storage stacks," *Frontiers in High Performance Computing*, vol. 3, Aug. 2025, doi: 10.3389/fhpcp.2025.1393936.

[8] W. Yang, R. Fu, M. B. Amin, and B. Kang, "The impact of modern AI in metadata management," *Human-Centric Intelligent Systems*, vol. 5, no. 3, pp. 323–350, Jul. 2025, doi: 10.1007/s44230-025-00106-5.

[9] S. Gore, S. Hamsa, S. Roychowdhury, G. Patil, S. Gore, and S. Karmode, "Augmented intelligence in machine learning for cybersecurity: Enhancing threat detection and human-machine collaboration," in *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, Aug. 2023, pp. 638–644, doi: 10.1109/ICAISS58487.2023.10250514.

[10] D. Miller, "Apply metadata tags to files with Tagit," *Macworld*. https://www.macworld.com/article/218684/apply_metadata_tags_to_files_with_tagit.html. (Accessed Jul. 05, 2025).

[11] J. Villamar *et al.*, "Metadata practices for simulation workflows," *Scientific Data*, vol. 12, no. 1, Jun. 2025, doi: 10.1038/s41597-025-05126-1.

[12] P. Petit and N. Vuillerme, "Datagraphy: Toward a systematic approach to dataset discovery," *GigaScience*, vol. 14, Jan. 2025, doi:

10.1093/gigascience/giaf134.
[13] H. Sim, Y. Kim, S. S. Vazhkudai, G. R. Vallée, S.-H. Lim, and A. R. Butt, "Tagit: An integrated indexing and search service for file systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2017, pp. 1–12, doi: 10.1145/3126908.3126929.
[14] W. Elouataoui, "AI-driven frameworks for enhancing data quality in big data Ecosystems: Error_detection, correction, and metadata integration," *arXiv*, May 2024, doi: 10.48550/arXiv.2405.03870.
[15] S. Gore, D. D. Jadhav, M. E. Ingale, S. Gore, and U. Nanavare, "Leveraging BERT for next-generation spoken language understanding with joint intent classification and slot filling," in *2023 International Conference on Advanced Computing Technologies and Applications (ICACTA)*, Oct. 2023, pp. 1–5, doi: 10.1109/ICACTA58201.2023.10393437.
[16] D. Boukraa, M. Bala, and S. Rizzi, "Metadata management in data lake environments: A survey," *Journal of Library Metadata*, vol. 24, no. 4, pp. 215–274, Oct. 2024, doi: 10.1080/19386389.2024.2359310.
[17] S. Ghimire *et al.*, "MIDAS: Adaptive proxy middleware for mitigating metadata hotspots in HPC I/O at scale," in *2025 lEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov. 2025, pp. 1–8, doi: 10.1109/CloudCom67567.2025.11331379.
[18] P. Poudel *et al.*, "A scalable framework for heterogeneous environmental data management using smart data pipeline," in *Practice and Experience in Advanced Research Computing 2025: The Power of Collaboration*, Jul. 2025, pp. 1–9, doi: 10.1145/3708035.3736017.
[19] D. Manno *et al.*, "GUFI: Fast, secure File system metadata search for both privileged and unprivileged users," in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2022, pp. 1–14, doi: 10.1109/SC41404.2022.00062.
[20] F. Ma, Y. Chen, Y. Zhou, Z. Yan, H. Sun, and Y. Jiang, "Finding metadata inconsistencies in distributed file systems via cross-node operation modeling," in *SEC '25: Proceedings of the 34th USENIX Conference on Security Symposium*, 2025, pp. 7525–7543.
[21] A. Khan, T. Kim, H. Byun, and Y. Kim, "SciSpace: A scientific collaboration workspace for geo-distributed HPC data centers," *Future Generation Computer Systems*, vol. 101, pp. 398–409, Dec. 2019, doi: 10.1016/j.future.2019.06.006.
[22] W. Zhang, S. Byna, H. Tang, B. Williams, and Y. Chen, "MIQS: Metadata indexing and querying service for self-describing file formats," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2019, pp. 1–24, doi: 10.1145/3295500.3356146.
[23] P. Cheng, Y. Wang, Y. Lu, Y. Du, and Z. Chen, "UniIndex: An index and query middleware for parallel file systems," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 9, May 2020, doi: 10.1002/cpe.5609.
[24] S. Dill *et al.*, "A case for automated large-scale semantic annotation," *Journal of Web Semantics*, vol. 1, no. 1, pp. 115–132, Dec. 2003, doi: 10.1016/j.websem.2003.07.006.
[25] C. Biardzki and T. Ludwig, "Analyzing metadata performance in distributed file systems," in *Parallel Computing Technologies (PaCT 2009)*, 2009, pp. 8–18, doi: 10.1007/978-3-642-03275-2_2.
[26] H. K. Omar and A. K. Jumaa, "Distributed big data analysis using spark parallel data processing," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, pp. 1505–1515, Jun. 2022, doi: 10.11591/eei.v11i3.3187.
[27] J. Sun, S. Li, J. Xu, and J. Huang, "The security war in file systems: An empirical study from a vulnerability-centric perspective," *ACM Transactions on Storage*, vol. 19, no. 4, pp. 1–26, Nov. 2023, doi: 10.1145/3606020.
[28] H. E. Khodke, M. Bhalerao, S. N. Gunjal, S. Nirmal, S. Gore, and B. J. Dange, "An intelligent approach to empowering the research of biomedical machine learning in medical data analysis using PALM," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 10s, pp. 429–436, 2023.
[29] "The 1000 genomes project consortium," *International Genome Sample Resource*. https://www.internationalgenome.org/data/. (Accessed Jan. 16, 2026).
[30] "CMIP6: Coupled Model Intercomparison Project Phase 6," *Lawrence Livermore National Laboratory*. https://pcmdi.llnl.gov/CMIP6. (Accessed Jan. 16, 2026).

BIOGRAPHIES OF AUTHORS

Sonali Vidhate is an Assistant Professor in Computer Engineering. She holds a Master's degree in Computer Science and is pursuing a Ph.D. Her research interests include distributed file systems, metadata indexing, and information retrieval. She has published research articles in peer-reviewed journals. She can be contacted at email: sonali.vidhate878@gmail.com.

Dr. Pankaj Dashore is an Associate Professor in Computer Engineering. He holds a Ph.D. degree in Computer. His research interests include distributed file systems, metadata indexing, and information retrieval. He has published research articles in peer-reviewed journals. He can be contacted at email: dashorepankaj@gmail.com.