Heisenpub

Volume 9 Issue 7
Journal: Reviews on Internet of Things (IoT), Cyber-Physical Systems, and Applications

# Building Scalable Data Lakes in the Cloud for Big Data Integration: Utilizing Amazon S3 and Apache Hadoop

Ahmad Faizal[†,*]

[†] Universiti Malaysia Pahang, Faculty of Computing,
Lebuhraya Tun Razak, Gambang, 26300 Kuantan, Pahang, Malaysia

Abstract. Building scalable data lakes in the cloud involves orchestrating a wide array of advanced computational and storage techniques to ensure robust and flexible handling of massive, heterogeneous datasets. This work presents a systematic approach for integrating large-scale data processes in cloud-based environments, with special attention given to the interplay between Amazon S3 and Apache Hadoop. Emphasis is placed on infrastructure design to maximize throughput, maintain reliability, and enable seamless elasticity. Key considerations span metadata management, distributed resource allocation, data partitioning, and fault-tolerant mechanisms that collectively uphold consistent performance under fluctuating workloads. Advanced mathematical modeling of resource consumption, concurrency controls, and data transfer rates is employed to elucidate optimal system configurations, while intricate scheduling paradigms and partitioning schemes are proposed to cater to evolving demands. Analytical formulations evaluate the role of structured transformations, batch processing, and real-time streaming within a unified architectural stack, ensuring minimal data latency and reduced operational overhead. By systematically addressing scalability requirements and performance trade-offs, this work provides a foundation for constructing a resilient data lake that leverages the raw object storage capabilities of Amazon S3 and the distributed processing power of Apache Hadoop. The proposed strategies enable comprehensive integration of data sources and facilitate efficient exploration of large-scale data for advanced analytic workflows.

## 1. Introduction

The proliferation of heterogeneous data sources in enterprise and research contexts necessitates infrastructure frameworks capable of accommodating both structured and unstructured formats on a massive scale [1]. Cloud-based data lakes have emerged as a viable strategy for delivering storage elasticity, operational flexibility, and cost optimization under intense data deluge. Exploiting scalable object stores such as Amazon S3 offers the advantage of seamless expansion without cumbersome capacity planning [2]. Meanwhile, distributed processing technologies like Apache Hadoop introduce parallel computation, robust fault tolerance, and a modular architecture conducive to multi-tenant data analysis.

The convergence of these paradigms into a holistic environment facilitates rapid data ingestion, transformation, and curation, thereby empowering organizations to derive data-driven insights. [3]

In many operational scenarios, data ingestion is highly dynamic and subject to unpredictable spikes in volume or velocity. A robust data lake must incorporate intelligent load balancing to accommodate uneven data distribution and spatiotemporal clustering of access patterns [4]. Concurrently, computational tasks that transform, cleanse, or enrich the data must be carefully choreographed to avoid system bottlenecks. Sophisticated scheduling algorithms, distributed file system abstractions, and advanced concurrency control mechanisms underpin the performance and availability of these pipelines. [5]

The fundamental building blocks for scalable data lakes center on cloud-native design principles, which prioritize elasticity, modularity, and shared-nothing execution models. Amazon S3 acts as the primary storage substrate, exploiting object-based organization, versioning, and replication. Apache Hadoop orchestrates batch data processing using MapReduce or more advanced execution engines, all while interfacing seamlessly with S3 for data input and output [6], [7]. Orchestration layers, query engines, and metadata catalogs round out the architecture, ensuring discoverability of data assets and optimal query planning across diverse workloads. This integrated environment must not only handle varied data types but also ensure reliable data retention, lineage tracking, and security compliance. [8]

Throughout the discussion, various mathematical models are interwoven to illustrate system behaviors, performance trade-offs, and scaling strategies. The conversation extends to topics such as the equilibrium analysis of resource-limited queues, the spectral decomposition of adjacency matrices in distributed worker assignment, and the direct application of partition functions for load distribution [9]. Though these concepts are deeply technical, they illuminate strategies for dimensioning capacity and selecting optimal data partitioning techniques. By unifying diverse computational paradigms within a single platform, cloud-hosted data lakes serve as a linchpin for big data integration, catalyzing innovation in analytics, machine learning, and beyond. [10]

## 2. Foundations of Cloud-Based Data Lakes

The construction of a cloud-based data lake hinges on a set of core principles. First among these is a disaggregated approach to storage and computation, where the cloud-based object store decouples data persistence from the processing layer [11]. This allows scaling of compute resources independently from data storage capacity, which is essential for cost-effective utilization of on-demand cloud instances. Amazon S3 acts as the reference example due to its near-infinite capacity, high durability via replication across availability zones, and event-based integration capabilities for real-time triggers.

In designing the logical structure of the data lake, there is often a need for a hierarchical partitioning scheme, which may involve time-based partitions or domain-specific directory

layouts [12]. Such partitions influence data retrieval patterns, caching strategies, and micro-batch ingestion. A mathematical treatment of partition optimization might consider the minimization of average query latency, expressed by an objective function [13], [14]

$$\min_{P \in \mathcal{P}} \sum_{i=1}^{n} \alpha_i \, \tau_i(P),$$

where $\alpha_i$ is the weight assigned to the $i$-th query or workload, and $\tau_i(P)$ denotes the latency associated with accessing the data partition indexed by $P$. This formulation highlights the trade-off between fine-grained partitioning, which improves query pruning, and coarse-grained partitioning, which reduces overhead in metadata management. [15]

Once data is placed in cloud object storage, metadata management is paramount for discoverability and efficient query planning. Centralized metadata repositories track schema information, data lineage, and partition boundaries [16]. The complexity of such repositories grows significantly as data volume expands in both size and dimensionality [17]. This can be understood via the concept of an evolving graph $G = (V, E)$ in which vertices represent data entities and edges encode lineage or transformation dependencies. As more transformations occur, the graph grows in edge cardinality, potentially leading to a combinatorial explosion in metadata [18]. Minimizing the cost of graph traversal, expressed as

$$\min \sum_{(v_i, v_j) \in E} f(v_i, v_j),$$

where $f$ is a function modeling the traversal time between related data vertices, becomes a key challenge for efficient metadata navigation [19]. This abstracted view underscores the requirement for robust indexing, caching, and summarization of metadata to prevent performance degradation under high query concurrency.

Data ingestion frameworks must also be carefully designed to handle diverse workloads, from micro-batches of streaming data to massive nightly ingestion tasks [20]. Each ingestion job interacts with Amazon S3's infrastructure, which may impose specific constraints related to object creation rates and eventual consistency models. In a multi-tenant scenario, system architects often rely on concurrency control algorithms to regulate ingestion pipelines [21]. A typical concurrency management approach employs a system of ordinary differential equations to describe the queue length of ingestion tasks over time:

$$\frac{dq(t)}{dt} = \lambda(t) - \mu(t)\Phi\big(q(t)\big),$$

where $\lambda(t)$ is the arrival rate of data ingestion requests, $\mu(t)$ reflects the service rate tied to the ingestion compute layer, and $\Phi$ is a function capturing nonlinear feedback effects [22]. Stability analysis of this system can provide insights into how quickly the data lake ingestion pipeline converges to a steady-state operating point under varying load profiles.

By grounding the data lake in a flexible, cloud-native paradigm, organizations gain the agility to spin up or tear down processing resources on demand. The next step is to integrate a distributed processing framework, specifically Apache Hadoop, to harness parallel computation [23]. This integration imposes additional architectural constraints and opportunities, especially concerning data transfer rates, shuffle boundaries, and the alignment of file splits with the underlying partition strategy. An appreciation of these constraints sets the stage for deeper exploration of methods to achieve robust and scalable performance as the data lake grows. [24]

## 3. Architectural Considerations for Scalability

A fundamental goal for a cloud-based data lake is to scale seamlessly in response to fluctuations in data volume, velocity, and variety. Scalability encompasses both horizontal scaling—adding more worker nodes to a Hadoop cluster or more ingest pipelines—and vertical scaling, which might involve employing more powerful compute instances or optimizing I/O throughput [25]. The guiding principle is to align resource provisioning with real-time workload characteristics, minimizing idling during lulls and preventing saturation during peak loads.

When integrating Apache Hadoop with Amazon S3, attention must be paid to read and write I/O patterns [26]. Hadoop's native file system abstractions can be mapped onto cloud object storage, but fundamental differences persist. Object stores generally lack the notion of in-place file mutation, which can complicate certain big data workloads that rely on append operations [27]. In many scenarios, a best practice is to write data to temporary locations within S3, and upon job completion, perform a rename operation. This approach, though simple in concept, may introduce additional overhead. The performance ramifications can be captured in a piecewise-defined cost function for job completion time: [28]

$$T_{\text{job}}(n) = \begin{cases} c_1 n + c_2 & \text{for small} \quad n, \\ c_3 \log(n) + c_4 & \text{for large} \quad n, \end{cases}$$

where $n$ is the size of the dataset processed, and the constants $c_1, c_2, c_3$, and $c_4$ relate to network bandwidth, rename overhead, and parallelization inefficiencies. This simplified model helps reason about cost breakpoints and guides decisions on how to chunk and organize data for parallel ingestion or transformation. [29]

Allocating computational resources within a Hadoop ecosystem typically entails decisions about YARN container sizing, queue allocation policies, and scheduler configurations. Mathematical modeling of resource allocation can often be viewed through the lens of integer programming, where one might solve an optimization problem: [30]

$$\min \sum_{j=1}^{m} \gamma_j u_j$$

subject to capacity and scheduling constraints, where $u_j$ denotes resource usage by job $j$ and $\gamma_j$ is the corresponding cost coefficient. This approach allows for dynamic assignment of tasks to available worker nodes, ensuring that the cluster remains balanced while minimizing total resource consumption. [31]

To further augment scalability, fault tolerance is critical. Cloud networks can exhibit transient errors, or ephemeral compute instances may fail without warning. Hadoop's resilience strategy hinges on replication of intermediate data and speculative execution of tasks that lag behind [32]. Probabilistic models for system reliability can be deployed to evaluate the mean time to data loss or the expected time to complete a job under varying fault conditions. For instance, consider a Markov chain with states corresponding to the number of failed nodes and transitions governed by failure and recovery rates [33]. The stationary distribution of this chain yields insights into the probability of cluster degradation impacting performance. This ensures that the data lake, with its indefinite lifespan, remains robust even under adverse conditions. [34]

When orchestrating a data lake at enterprise scale, operational governance must also be integrated, including fine-grained access controls and encryption at rest. While these policy aspects might not initially appear mathematically grounded, the overhead induced by encryption and decryption processes can be analyzed using standard cryptographic performance models, such as measuring the computational cost in cycles per byte [35]. Ultimately, strategic selection of encryption ciphers with hardware acceleration can minimize overhead, making such security mechanisms feasible for large-scale data operations.

## 4. Integration with Amazon S3 and Apache Hadoop

The interplay between Amazon S3 and Apache Hadoop constitutes the core mechanism by which large datasets are stored, accessed, and transformed in a cloud-based data lake [36]. The Hadoop S3 filesystem connector enables job inputs to be read directly from S3, while outputs can be staged in S3 or ephemeral storage. The challenge lies in ensuring consistent, predictable performance when scaling to thousands of concurrent queries or large batch-processing jobs.

One area that warrants detailed exploration is the read throughput from S3 to Hadoop workers [37]. The concurrency limits for a single S3 bucket could potentially cap overall performance if insufficient parallel streams are utilized. In practice, each mapper or reducer task can establish multiple TCP connections to read from S3, generating an aggregate throughput that may approach the data link speed [38]. However, there are cases where the overhead of opening many connections can induce diminishing returns or lead to connection throttling. A rigorous analysis might treat each connection as a queueing station, with the arrival rate of data request packets $\lambda$ and a service rate $\mu$ [39], [40]. The expected time in the system for each request is given by the standard M/M/1 formula:

$$E(T) = \frac{1}{\mu - \lambda},$$

illustrating that as $\lambda$ approaches $\mu$, the system becomes highly sensitive to even small load changes [41]. A large-scale cluster design must avoid such a saturation regime by balancing the concurrency level across tasks.

Similarly, when writing output back to S3, the cost model must account for both network transit time and S3 PUT or multi-part upload operations. Multi-part uploads allow parallelization of the write process, reducing total completion time [42]. However, the overhead of coordinating multiple parts introduces synchronization points that may be modeled via a fork-join queue, in which tasks split into parallel subtasks and then rejoin. The wait time at the join point can become a bottleneck if the distribution of subtask completion times is skewed [43]. One might employ the fork-join queue approximation to assess expected completion time:

$$E(T_{\text{fork-join}}) \approx \frac{1}{m} \sum_{i=1}^{m} E(T_i) + \zeta\big(\sigma^2(T_i)\big),$$

where $m$ is the number of subtasks, $E(T_i)$ is the mean completion time for the $i$-th subtask, and $\zeta$ is a function mapping subtask time variance to extra delay at the join point [44]. These models inform decisions on how to break down large writes into multiple parts, particularly for high-throughput analytics jobs.

Beyond simple MapReduce, the Hadoop ecosystem encompasses more sophisticated engines for SQL-like querying, iterative machine learning, and stream processing [45]. Interacting with S3 from these engines involves a similar set of challenges in concurrency, consistency, and partition design. Even subtle changes in query engines, such as the introduction of pushdown predicates or vectorized I/O, can significantly alter the performance profile when reading from S3 [46]. Monitoring and tuning these interactions require an integrated view of the entire pipeline, with special attention to ephemeral states in the cluster and the ephemeral nature of dynamic scaling decisions in the cloud. An optimal integration strategy, therefore, is one that continuously adapts data partitioning, concurrency levels, and job scheduling configurations to the workload's real-time demands.

## 5. Advanced Data Ingestion and Processing Strategies

Cloud-based data lakes frequently contend with mixed workloads that demand both near-real-time ingestion pipelines and batch-oriented transformations [47]. Achieving consistent performance across this spectrum can require advanced ingestion orchestration layers. One possible approach uses ephemeral compute clusters spun up to handle ingestion spikes, dissolving them after the workload subsides [48]. The ephemeral nature of these clusters adds complexity in provisioning, data locality, and the management of intermediate states.

Mathematical models can clarify how ephemeral cluster usage impacts overall cost and responsiveness [49]. Consider a function $C(N, T)$ representing the total cost for provisioning $N$ nodes for a duration of $T$. In a pay-as-you-go environment, $C$ might scale linearly with $N \times T$, though the elasticity overhead arises from spin-up times and potential ephemeral node underutilization [50]. If data arrival follows a non-homogeneous Poisson process with

rate $\lambda(t)$, the scheduling of ephemeral clusters can be mapped to an optimal control problem in which one seeks to minimize the integral of cost subject to a service level constraint:

$$\min_{N(t)} \int_0^{\Omega} \Big[ \alpha \, N(t) + \beta \, w(t) \Big] dt,$$

where $\alpha$ weights the cost of provisioning nodes, $\beta$ weights the penalty for queueing delays, and $w(t)$ is a function describing backlog [51]. By examining the optimal control solution, architects can derive thresholds for when it is most cost-effective to provision ephemeral nodes versus relying on baseline capacity.

Processing strategies in a data lake also extend beyond batch. There is often a need for streaming transformations that apply real-time analytics or machine learning inference [52], [53]. Stream processors might be layered on top of Apache Hadoop or integrated through an event-driven architecture. These real-time systems must handle event ordering, exactly-once semantics, and out-of-order arrivals [54]. When bridging streaming layers with S3 storage, consistent checkpointing procedures are critical to maintain fault tolerance. The checkpoint overhead can be analyzed by representing each checkpoint as an independent Bernoulli trial of success or failure, with probability $p$ [55]. Over a sequence of streaming micro-batches, one seeks to maximize the probability of a consistent state:

$$P(\text{consistent state}) = \prod_{k=1}^{m} p_k,$$

where $p_k$ is the success probability of the $k$-th checkpoint [56]. The introduction of ephemeral cluster nodes can further complicate checkpointing, as short-lived nodes may vanish mid-stream. In practice, strategies that store checkpoints directly in S3 help ensure the durability of state beyond the lifetime of any single compute resource.

Data transformations, especially those dealing with large-scale joins or multi-stage aggregations, benefit from adaptive query execution techniques that reorder tasks at runtime based on observed data statistics [57]. Such techniques can exploit partial histograms derived from earlier map tasks to refine load balancing in subsequent stages. This approach can be examined with combinatorial optimization, in which one tries to maximize a performance metric subject to constraints on shuffle volumes or memory footprints [58]. Symbolically, one might define a bipartite matching problem with sets representing tasks and nodes, seeking a matching that balances load while adhering to memory constraints:

$$\max \sum_{(t,n) \in M} \delta_{(t,n)},$$

where $\delta_{(t,n)}$ is a performance gain metric if task $t$ is assigned to node $n$. A feasible matching $M$ must satisfy capacity constraints for each node [59]. Solving such a matching in real time, or approximating the solution, underpins the advantage of dynamic, data-aware scheduling.

Multitenancy is another dimension of complexity in large data lakes, where multiple organizational units or teams share the same environment [60]. This can lead to contention in both storage and compute resources, impacting throughput. Applying fair scheduling or capacity scheduling requires shaping traffic from each tenant to ensure that large, compute-intensive tasks do not preclude smaller, latency-sensitive queries from completing in a timely manner [61]. By employing advanced scheduling algorithms and carefully calibrated concurrency controls, the system can simultaneously handle a diverse set of analytics tasks, ensuring consistent performance and resource fairness.

## 6. Performance Evaluation and Mathematical Modeling

Evaluating the performance of a cloud-based data lake that leverages S3 and Hadoop requires defining standardized metrics and leveraging mathematical modeling to interpret the impact of architectural choices. Typical metrics include query latency, throughput, scalability factor, cost-efficiency, and fault tolerance [62], [63]. The interplay among these metrics is complex, and performance modeling aids in predicting system behavior under future load scenarios.

Benchmarking is often conducted by running a suite of workloads that simulate real-world usage patterns [64], [65]. For instance, synthetic data might be generated following a known distribution, such as a Pareto or lognormal distribution, to mirror long-tail file sizes. The concurrency level of queries can be systematically increased while monitoring response times [66]. One can construct a time series of performance measurements $\{T_i\}$, from which to compute descriptive statistics:

$$\overline{T} = \frac{1}{N}\sum_{i=1}^{N} T_i, \quad \sigma(T) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(T_i - \overline{T}\right)^2}.$$

In a properly dimensioned data lake, the mean latency $\overline{T}$ should remain nearly constant under moderate concurrency, with only mild increases until approaching the cluster's capacity limit.

Mathematical modeling provides additional granularity, facilitating a deeper exploration of how performance metrics scale. For example, modeling concurrency with a multi-class queueing network can capture distinct classes of queries, each with unique arrival rates $\lambda_c$ and service requirements [67]. Such a network might consist of various service stations corresponding to different Hadoop resources or distinct S3 buckets. By analyzing the network's steady-state probabilities, one can infer utilization levels, queue lengths, and overall latency distributions [68]. A typical multi-class network might be defined by a routing matrix $R$, where $R_{ij}$ is the probability that a request moves from station $i$ to station $j$. The flow balance conditions in steady state yield a system of linear equations:

$$\Lambda_j = \sum_{i=1}^{k} \Lambda_i \, R_{ij},$$

where $\Lambda_j$ denotes the effective arrival rate to station $j$. Solving these equations sheds light on which stations are likely bottlenecks [69]. Adjusting capacity or concurrency at these stations can drastically alter end-to-end performance.

Scalability can be further examined via strong scaling (how performance improves when resources increase, while data size is held constant) and weak scaling (how performance behaves when data size grows proportionally with resources) [70]. Data lake architectures often aim for near-linear scaling, though in practice diminishing returns appear beyond certain resource thresholds due to overhead in coordination, shuffle, or synchronization. This phenomenon can be captured by a function akin to Amdahl's Law: [71]

$$S_p = \frac{1}{f + \frac{1-f}{p}},$$

where $S_p$ is the speedup for $p$ parallel workers, and $f$ is the fraction of work that cannot be parallelized. Although originally formulated for CPU-bound parallelism, analogous reasoning applies to the distributed environment in which network overhead, shuffle cost, and single-threaded metadata services limit the fraction of code that scales perfectly. [72]

Modeling reliability also features in performance assessment. A robust data lake should maintain operational continuity even under node failures or partial network outages. Stochastic processes such as birth-death chains can describe the rate of node arrivals (recoveries) and departures (failures) [73]. Metrics of interest include the mean time between system reconfigurations and the probability that a job completes before encountering a critical fault. Evaluations of these metrics can inform design decisions about replication levels for intermediate data, as well as threshold policies for speculative task re-execution. [74]

Overall, performance evaluation couples empirical measurements from a well-instrumented data lake with analytically grounded models. This synthesis enables the design of advanced architectures that respond flexibly to the ephemeral nature of cloud environments, the evolving complexity of big data workloads, and the strict operational requirements of large organizations. [75]

## 7. Conclusion

Building a scalable data lake in the cloud for big data integration requires harmonizing storage and processing technologies in a manner that embraces elasticity, fault tolerance, and performance optimization. By leveraging Amazon S3's capacity, durability, and global accessibility, organizations gain a robust foundation for storing expansive datasets [76]. When coupled with Apache Hadoop's distributed processing engine, this integrated architecture addresses the challenges of large-scale data ingestion, transformation, and analysis within a unified framework. Architectural designs focusing on disaggregated storage and compute allow for independent scaling of resources, while advanced data partitioning strategies, concurrency controls, and scheduling algorithms ensure efficient utilization of those resources over time.

Mathematical models illuminate the nuanced relationships among workload concurrency, infrastructure provisioning, and throughput objectives, providing a theoretical lens through which one can evaluate and refine system choices. Concurrency models capture how ingestion or query tasks queue and propagate through the data lake, while optimization frameworks guide dynamic resource allocation to align closely with real-time demands. Techniques such as fork-join approximations explain the potential bottlenecks arising from multi-part uploads, whereas multi-class queueing networks reveal how different query types compete for shared resources [77]. The breadth and depth of these analytical approaches highlight the intricate demands placed on cloud-based data lakes that must handle petabyte-scale data, unpredictable data arrival patterns, and complex analytics pipelines.

In merging advanced computing paradigms with cloud-native storage, the resulting data lake architecture becomes an adaptive ecosystem capable of accommodating both structured and unstructured data at extreme scale [78]. This adaptability extends to accommodating diverse analytical requirements, from low-latency streaming ingestion to high-throughput batch transformations. The dynamic interplay between ephemeral compute resources and permanent object storage enables cost-effective operations, since capacity can be tailored precisely to workload intensity [79]. Integrating advanced cryptographic routines and governance mechanisms further ensures that the environment remains secure and compliant.

The strategic combination of cloud-based object stores and distributed processing engines supports an iterative cycle of data exploration, iterative modeling, and operational analytics [80]. As organizations continue to expand their data footprints, robust data lakes underpinned by frameworks such as Amazon S3 and Apache Hadoop will remain pivotal in supporting high-volume, high-velocity data analysis. Continual advances in storage abstractions, execution engines, and scheduling algorithms promise even greater efficiency and scalability, paving the way for future deployments that exploit real-time machine learning, semantic data catalogs, and next-generation data formats. Ultimately, the synergistic relationship between scalable cloud storage and distributed computation represents a transformative paradigm for big data integration, unlocking the potential for sustained innovation across industry and research domains. [81]

## References

[1] S. Alaimo, A. D. Maria, D. Shasha, A. Ferro, and A. Pulvirenti, "Tacitus: Transcriptomic data collector, integrator, and selector on big data platform," *BMC bioinformatics*, vol. 20, no. 9, pp. 366–, Nov. 22, 2019. DOI: 10.1186/s12859-019-2912-4.

[2] T.-W. Um, E. Lee, G. M. Lee, and Y. Yoon, "Design and implementation of a trust information management platform for social internet of things environments.," *Sensors (Basel, Switzerland)*, vol. 19, no. 21, pp. 4707–, Oct. 29, 2019. DOI: 10.3390/s19214707.

[3] S. U. Zuliana and A. Perperoglou, "Two dimensional smoothing via an optimised whittaker smoother," *Big Data Analytics*, vol. 2, no. 1, pp. 6–, Mar. 13, 2017. DOI: 10.1186/s41044-017-0021-9.

[4] T. G. Myers, P. N. Ramkumar, B. F. Ricciardi, K. L. Urish, J. Kipper, and C. Ketonis, "Artificial intelligence and orthopaedics: An introduction for clinicians.," *The Journal of bone and joint surgery. American volume*, vol. 102, no. 9, pp. 830–840, Feb. 6, 2020. DOI: 10.2106/jbjs.19.01128.

[5] A. G. Rumson and S. H. Hallett, "Innovations in the use of data facilitating insurance as a resilience mechanism for coastal flood risk.," *The Science of the total environment*, vol. 661, pp. 598–612, Jan. 14, 2019. DOI: 10.1016/j.scitotenv.2019.01.114.

[6] K. Venkatesh, M. J. S. Ali, N. Nithiyanandam, and M. Rajesh, "Challenges and research disputes and tools in big data analytics," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 6s3, pp. 1949–1952, Nov. 22, 2019. DOI: 10.35940/ijeat.f1376.0986s319.

[7] R. Avula, "Strategies for minimizing delays and enhancing workflow efficiency by managing data dependencies in healthcare pipelines," *Eigenpub Review of Science and Technology*, vol. 4, no. 1, pp. 38–57, 2020.

[8] O. Adedugbe, E. Benkhelifa, R. Campion, F. N. Al-Obeidat, A. B. Hani, and U. Jayawickrama, "Leveraging cloud computing for the semantic web: Review and trends," *Soft Computing*, vol. 24, no. 8, pp. 5999–6014, Nov. 29, 2019. DOI: 10.1007/s00500-019-04559-2.

[9] H. Hassani, X. Huang, and M. Ghodsi, "Big data and causality," *Annals of Data Science*, vol. 5, no. 2, pp. 133–156, Aug. 1, 2017. DOI: 10.1007/s40745-017-0122-3.

[10] Y. Duan, N. Wang, and J. Wu, "Accelerating dag-style job execution via optimizing resource pipeline scheduling," *Journal of Computer Science and Technology*, vol. 37, no. 4, pp. 852–868, Jul. 30, 2022. DOI: 10.1007/s11390-021-1488-4.

[11] S. S. Farley, A. Dawson, S. Goring, and J. W. Williams, "Situating ecology as a big-data science: Current advances, challenges, and solutions," *BioScience*, vol. 68, no. 8, pp. 563–576, Jul. 18, 2018. DOI: 10.1093/biosci/biy068.

[12] E. B. Molder, S. F. Schenkein, A. E. McConnell, K. K. Benedict, and C. L. Straub, "Landsat data ecosystem case study: Actor perceptions of the use and value of landsat," *Frontiers in Environmental Science*, vol. 9, Feb. 4, 2022. DOI: 10.3389/fenvs.2021.805174.

[13] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, pp. 1–30, Nov. 1, 2018. DOI: 10.1186/s40537-018-0151-6.

[14] M. Kansara, "A comparative analysis of security algorithms and mechanisms for protecting data, applications, and services during cloud migration," *International Journal of Information and Cybersecurity*, vol. 6, no. 1, pp. 164–197, 2022.

[15] A. S. Kaseb, A. Mohan, Y. Koh, and Y.-H. Lu, "Cloud resource management for analyzing big real-time visual data from network cameras," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 935–948, Oct. 1, 2019. DOI: 10.1109/tcc.2017.2720665.

[16] B. Zhang, L. Zhu, Z. Pei, *et al.*, "A framework for remote interaction and management of home care elderly adults," *IEEE Sensors Journal*, vol. 22, no. 11, pp. 11 034–11 044, Jun. 1, 2022. DOI: 10.1109/jsen.2022.3170295.

[17] R. Avula, "Addressing barriers in data collection, transmission, and security to optimize data availability in healthcare systems for improved clinical decision-making and analytics," *Applied Research in Artificial Intelligence and Cloud Computing*, vol. 4, no. 1, pp. 78–93, 2021.

[18] Z. X. Hou and Y. Xiao, "Special issue on big data for iot cloud computing convergence," *Web Intelligence*, vol. 17, no. 2, pp. 101–103, Apr. 8, 2019. DOI: 10.3233/web-190404.

[19] D. M. Kotliar, "Data orientalism: On the algorithmic construction of the non-western other," *Theory and Society*, vol. 49, no. 5, pp. 919–939, Aug. 27, 2020. DOI: 10.1007/s11186-020-09404-2.

[20] M. Khanna, B. M. Gramig, E. H. DeLucia, X. Cai, and P. Kumar, "Harnessing emerging technologies to reduce gulf hypoxia," *Nature Sustainability*, vol. 2, no. 10, pp. 889–891, Sep. 23, 2019. DOI: 10.1038/s41893-019-0381-4.

[21] C. Kwan, L. Hagen, B. Chou, *et al.*, "Simple and effective cloud- and shadow-detection algorithms for landsat and worldview images," *Signal, Image and Video Processing*, vol. 14, no. 1, pp. 125–133, Jul. 25, 2019. DOI: 10.1007/s11760-019-01532-2.

[22] J. Hu and Y. Zhang, "Discovering the interdisciplinary nature of big data research through social network analysis and visualization," *Scientometrics*, vol. 112, no. 1, pp. 91–109, May 8, 2017. DOI: 10.1007/s11192-017-2383-1.

[23] R. Ranchal, P. R. Bastide, X. Wang, *et al.*, "Disrupting healthcare silos: Addressing data volume, velocity and variety with a cloud-native healthcare data ingestion service," *IEEE journal of biomedical and health informatics*, vol. 24, no. 11, pp. 3182–3188, Nov. 4, 2020. DOI: 10.1109/jbhi.2020.3001518.

[24] A. Qayyum, A. Ijaz, M. Usama, *et al.*, "Securing machine learning in the cloud: A systematic review of cloud machine learning security.," *Frontiers in big data*, vol. 3, pp. 587 139–587 139, Nov. 12, 2020. DOI: 10.3389/fdata.2020.587139.

[25] L. Jiang, L. Liu, J. Yao, and L.-L. Shi, "A hybrid recommendation model in social media based on deep emotion analysis and multi-source view fusion," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–16, Oct. 7, 2020. DOI: 10.1186/s13677-020-00199-2.

[26] M. A. Gulzar, M. Interlandi, X. Han, M. Li, T. Condie, and M. Kim, "Socc - automated debugging in data-intensive scalable computing," *Proceedings of the ... ACM Symposium on Cloud Computing [electronic resource] : SOCC ... ... SoCC (Conference)*, vol. 2017, pp. 520–534, Sep. 24, 2017. DOI: 10.1145/3127479.3131624.

[27] M. T. Patterson, N. Anderson, C. Bennett, *et al.*, "The matsu wheel: A reanalysis framework for earth satellite imagery in data commons," *International Journal of Data Science and Analytics*, vol. 4, no. 4, pp. 251–264, Mar. 30, 2017. DOI: 10.1007/s41060-017-0052-3.

[28] T. Hu, S. Wang, W. Luo, *et al.*, "Revealing public opinion towards covid-19 vaccines with twitter data in the united states: Spatiotemporal perspective.," *Journal of medical Internet research*, vol. 23, no. 9, e30854–, Sep. 10, 2021. DOI: 10.2196/30854.

[29] C. Ogle, D. Reddick, C. B. McKnight, *et al.*, "Named data networking for genomics data management and integrated workflows," *Frontiers in big data*, vol. 4, pp. 582 468–582 468, Feb. 15, 2021. DOI: 10.3389/fdata.2021.582468.

[30] K. Sheng, "Artificial intelligence in radiotherapy: A technological review.," *Frontiers of medicine*, vol. 14, no. 4, pp. 431–449, Jul. 29, 2020. DOI: 10.1007/s11684-020-0761-1.

[31] K. D. Strang, "Problems with research methods in medical device big data analytics," *International Journal of Data Science and Analytics*, vol. 9, no. 2, pp. 229–240, Feb. 5, 2019. DOI: 10.1007/s41060-019-00176-2.

[32] C. Pan, G. McInnes, N. A. Deflaux, *et al.*, "Cloud-based interactive analytics for terabytes of genomic variants data.," *Bioinformatics (Oxford, England)*, vol. 33, no. 23, pp. 3709–3715, Jul. 26, 2017. DOI: 10.1093/bioinformatics/btx468.

[33] A. Alekseev, S. Campana, X. Espinal, *et al.*, "On the road to a scientific data lake for the high luminosity lhc era," *International Journal of Modern Physics A*, vol. 35, no. 33, pp. 2 030 022–, Nov. 30, 2020. DOI: 10.1142/s0217751x20300227.

[34] M. Hassan, A. I. E. Desouky, M. M. Badawy, A. Sarhan, M. Elhoseny, and M. Gunasekaran, "Eot-driven hybrid ambient assisted living framework with naïve bayes–firefly algorithm," *Neural Computing and Applications*, vol. 31, no. 5, pp. 1275–1300, May 11, 2018. DOI: 10.1007/s00521-018-3533-y.

[35] R. T. Ilieva and T. McPhearson, "Social-media data for urban sustainability," *Nature Sustainability*, vol. 1, no. 10, pp. 553–565, Oct. 15, 2018. DOI: 10.1038/s41893-018-0153-6.

[36] B. B. Gupta, D. P. Agrawal, S. Yamaguchi, and M. Sheng, "Soft computing techniques for big data and cloud computing," *Soft Computing*, vol. 24, no. 8, pp. 5483–5484, Mar. 6, 2020. DOI: 10.1007/s00500-020-04766-2.

[37] C. Liu, J. Gao, Y. Li, H. Wang, and Z. Chen, "Studying gas exceptions in blockchain-based cloud applications," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–25, Jun. 30, 2020. DOI: 10.1186/s13677-020-00176-9.

[38] L. Bellatreche and S. Chakravarthy, "A special issue in extending data warehouses to big data analytics," *Distributed and Parallel Databases*, vol. 37, no. 3, pp. 323–327, Feb. 26, 2019. DOI: 10.1007/s10619-019-07262-1.

[39] S. Hosseinalipour, A. Nayak, and H. Dai, "Power-aware allocation of graph jobs in geo-distributed cloud networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 749–765, Apr. 1, 2020. DOI: 10.1109/tpds.2019.2943457.

[40] M. Kansara, "A structured lifecycle approach to large-scale cloud database migration: Challenges and strategies for an optimal transition," *Applied Research in Artificial Intelligence and Cloud Computing*, vol. 5, no. 1, pp. 237–261, 2022.

[41] Y. Ding and A. Javadi-Abhari, "Quantum and post-moore's law computing," *IEEE Internet Computing*, vol. 26, no. 1, pp. 5–6, Jan. 1, 2022. DOI: 10.1109/mic.2021.3133675.

[42] H. Riris, K. Numata, S. Wu, and M. E. Fahey, "The challenges of measuring methane from space with a lidar," *CEAS Space Journal*, vol. 11, no. 4, pp. 475–483, Sep. 14, 2019. DOI: 10.1007/s12567-019-00274-8.

[43] M. Mahdianpari, B. Salehi, F. Mohammadimanesh, *et al.*, "Big data for a big country: The first generation of canadian wetland inventory map at a spatial resolution of 10-m using sentinel-1 and sentinel-2 data on the google earth engine cloud computing platform," *Canadian Journal of Remote Sensing*, vol. 46, no. 1, pp. 15–33, Jan. 2, 2020. DOI: 10.1080/07038992.2019.1711366.

[44] P. Radanliev, D. D. Roure, R. Walton, *et al.*, "Covid-19 what have we learned? the rise of social machines and connected devices in pandemic management following the concepts of predictive, preventive and personalized medicine," *The EPMA journal*, vol. 11, no. 3, pp. 311–332, Jul. 30, 2020. DOI: 10.1007/s13167-020-00218-x.

[45] P. Treleaven, J. Barnett, A. Knight, and W. Serrano, "Real estate data marketplace," *AI and Ethics*, vol. 1, no. 4, pp. 445–462, Apr. 20, 2021. DOI: 10.1007/s43681-021-00053-4.

[46] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan, "Stochastic load balancing for virtual resource management in datacenters," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 459–472, Apr. 1, 2020. DOI: 10.1109/tcc.2016.2525984.

[47] A. A. Laghari, H. He, M. Shafiq, and A. Khan, "Application of quality of experience in networked services: Review, trend & perspectives," *Systemic Practice and Action Research*, vol. 32, no. 5, pp. 501–519, Oct. 17, 2018. DOI: 10.1007/s11213-018-9471-x.

[48] S. Zerdoumi, A. Q. M. Sabri, A. Kamsin, *et al.*, "Image pattern recognition in big data: Taxonomy and open challenges: Survey," *Multimedia Tools and Applications*, vol. 77, no. 8, pp. 10 091–10 121, Aug. 25, 2017. DOI: 10.1007/s11042-017-5045-7.

[49] S. Liu, L. Zhang, and B. Wang, "Individual diversity between interdependent networks promotes the evolution of cooperation by means of mixed coupling.," *Scientific reports*, vol. 9, no. 1, pp. 11 163–11 163, Aug. 1, 2019. DOI: 10.1038/s41598-019-47013-x.

[50] F. Lucivero, "Big data, big waste? a reflection on the environmental sustainability of big data initiatives.," *Science and engineering ethics*, vol. 26, no. 2, pp. 1009–1030, Dec. 23, 2019. DOI: 10.1007/s11948-019-00171-7.

[51] C. Li, J. Zhang, and H. Tang, "Replica-aware task scheduling and load balanced cache placement for delay reduction in multi-cloud environment," *The Journal of Supercomputing*, vol. 75, no. 5, pp. 2805–2836, Nov. 17, 2018. DOI: 10.1007/s11227-018-2695-9.

[52] S. Wang, X. Jiang, H. Tang, *et al.*, "A community effort to protect genomic data sharing, collaboration and outsourcing," *NPJ genomic medicine*, vol. 2, no. 1, pp. 33–33, Oct. 27, 2017. DOI: 10.1038/s41525-017-0036-1.

[53] R. Avula, "Assessing the impact of data quality on predictive analytics in healthcare: Strategies, tools, and techniques for ensuring accuracy, completeness, and timeliness in electronic health records," *Sage Science Review of Applied Machine Learning*, vol. 4, no. 2, pp. 31–47, 2021.

[54] P. B. Ross, J. Song, P. S. Tsao, and C. Pan, "Trellis for efficient data and task management in the va million veteran program.," *Scientific reports*, vol. 11, no. 1, pp. 23 229–, Dec. 1, 2021. DOI: 10.1038/s41598-021-02569-5.

[55] F. J. Clemente-Castello, B. Nicolae, R. Mayo, and J. C. Fernández, "Performance model of mapreduce iterative applications for hybrid cloud bursting," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 8, pp. 1794–1807, Aug. 1, 2018. DOI: 10.1109/tpds.2018.2802932.

[56] R. Abernathey, T. Augspurger, A. Banihirwe, *et al.*, "Cloud-native repositories for big scientific data," *Computing in Science & Engineering*, vol. 23, no. 2, pp. 26–35, Mar. 1, 2021. DOI: 10.1109/mcse.2021.3059437.

[57] Alasmari, W. Wang, T. Qin, and Y. Wang, "Proof of outsourced encryption: Cross verification of security service level agreement," *CCF Transactions on Networking*, vol. 3, no. 3, pp. 229–244, Nov. 24, 2020. DOI: 10.1007/s42045-020-00046-7.

[58] S. Yang, Z. Zhang, C. Zhao, X. Song, S. Guo, and H. Li, "Cnnpc: End-edge-cloud collaborative cnn inference with joint model partition and compression," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4039–4056, Dec. 1, 2022. DOI: 10.1109/tpds.2022.3177782.

[59] W. Lin, G. Peng, X. Bian, S. Xu, V. Chang, and Y. Li, "Scheduling algorithms for heterogeneous cloud environment: Main resource load balancing algorithm and time balancing algorithm," *Journal of Grid Computing*, vol. 17, no. 4, pp. 699–726, Nov. 19, 2019. DOI: 10.1007/s10723-019-09499-7.

[60] S. El-Sappagh, F. Ali, A. M. Hendawi, J.-H. Jang, and K. S. Kwak, "A mobile health monitoring-and-treatment system based on integration of the ssn sensor ontology and the

hl7 fhir standard," *BMC medical informatics and decision making*, vol. 19, no. 1, pp. 1–36, May 10, 2019. DOI: 10.1186/s12911-019-0806-z.

[61] V. Shankar, "Big data and analytics in retailing," *NIM Marketing Intelligence Review*, vol. 11, no. 1, pp. 36–40, Apr. 24, 2019. DOI: 10.2478/nimmir-2019-0006.

[62] S. Kaur, P. Bagga, R. Hans, and H. Kaur, "Quality of service (qos) aware workflow scheduling (wfs) in cloud computing: A systematic review," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 2867–2897, Nov. 3, 2018. DOI: 10.1007/s13369-018-3614-3.

[63] M. Kansara, "A framework for automation of cloud migrations for efficiency, scalability, and robust security across diverse infrastructures," *Quarterly Journal of Emerging Technologies and Innovations*, vol. 8, no. 2, pp. 173–189, 2023.

[64] Y. Himeur, M. Elnour, F. Fadli, *et al.*, "Ai-big data analytics for building automation and management systems: A survey, actual challenges and future perspectives.," *Artificial intelligence review*, vol. 56, no. 6, pp. 4929–5021, Oct. 15, 2022. DOI: 10.1007/s10462-022-10286-2.

[65] A. Sharma and K. M. Goolsbey, "Simulation-based approach to efficient commonsense reasoning in very large knowledge bases," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1360–1367.

[66] F. Stefanello, V. Aggarwal, L. S. Buriol, and M. G. C. Resende, "Hybrid algorithms for placement of virtual machines across geo-separated data centers," *Journal of Combinatorial Optimization*, vol. 38, no. 3, pp. 748–793, May 10, 2019. DOI: 10.1007/s10878-019-00411-3.

[67] A. Mison, G. Davies, and P. Eden, "Role of big tech in future cyber defence," *International Conference on Cyber Warfare and Security*, vol. 17, no. 1, pp. 583–590, Mar. 2, 2022. DOI: 10.34190/iccws.17.1.73.

[68] J. Yu, G. Zhu, H. Chen, L. Wang, and M. Xu, "Research on software architecture optimization of cloud computing data center based on hadoop," *IOP Conference Series: Materials Science and Engineering*, vol. 677, no. 4, pp. 042 015–, Dec. 1, 2019. DOI: 10.1088/1757-899x/677/4/042015.

[69] S. Koppad, A. B, G. V. Gkoutos, and A. Acharjee, "Cloud computing enabled big multi-omics data analytics," *Bioinformatics and biology insights*, vol. 15, pp. 11 779 322 211 035 921–, Jul. 28, 2021. DOI: 10.1177/11779322211035921.

[70] F. A. Khan, A.-u. Rahman, M. Alharbi, and Y. K. Qawqzeh, "Awareness and willingness to use phr: A roadmap towards cloud-dew architecture based phr framework," *Multimedia Tools and Applications*, vol. 79, no. 13, pp. 8399–8413, Sep. 25, 2018. DOI: 10.1007/s11042-018-6692-z.

[71] B. Muthu, C. B. Sivaparthipan, G. Manogaran, *et al.*, "Iot based wearable sensor for diseases prediction and symptom analysis in healthcare sector," *Peer-to-Peer Networking and Applications*, vol. 13, no. 6, pp. 2123–2134, Jan. 29, 2020. DOI: 10.1007/s12083-019-00823-2.

[72] R. Ranjan, Z. Li, M. Villari, Y. Liu, and D. Georgeakopoulos, "Software-driven big data analytics," *Computing*, vol. 102, no. 6, pp. 1409–1417, Jun. 5, 2020. DOI: 10.1007/s00607-020-00822-9.

[73] T. Li, L. Wang, Y. Ren, X. Li, J. Xia, and R. An, "An efficient method for meteorological nephogram recognition in cloud environment," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–10, Dec. 17, 2019. DOI: 10.1186/s13638-019-1611-1.

[74] M. Sellami, H. Mezni, M. S. Hacid, and M. M. Gammoudi, "Clustering-based data placement in cloud computing: A predictive approach," *Cluster Computing*, vol. 24, no. 4, pp. 3311–3336, Jun. 16, 2021. DOI: 10.1007/s10586-021-03332-1.

[75] O. C. Anejionu, Y. Sun, P. Thakuriah, A. McHugh, and P. Mason, "Great britain transport, housing, and employment access datasets for small-area urban area analytics.," *Data in brief*, vol. 27, pp. 104 616–104 616, Oct. 14, 2019. DOI: 10.1016/j.dib.2019.104616.

[76] Y. Wang, X. Tao, F. Zhao, B. Tian, and A. M. V. V. Sai, "Sla-aware resource scheduling algorithm for cloud storage," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–10, Jan. 3, 2020. DOI: 10.1186/s13638-019-1604-0.

[77] M. B. da Costa, L. M. A. L. D. Santos, J. L. Schaefer, I. C. Baierle, and E. O. B. Nara, "Industry 4.0 technologies basic network identification," *Scientometrics*, vol. 121, no. 2, pp. 977–994, Sep. 6, 2019. DOI: 10.1007/s11192-019-03216-7.

[78] Y. Xu, Y. Li, Z. Shen, *et al.*, "Parallel multiple instance learning for extremely large histopathology image analysis.," *BMC bioinformatics*, vol. 18, no. 1, pp. 360–360, Aug. 3, 2017. DOI: 10.1186/s12859-017-1768-8.

[79] A. Ndikumana, N. H. Tran, T. M. Ho, *et al.*, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1359–1374, Jun. 1, 2020. DOI: 10.1109/tmc.2019.2908403.

[80] A. Yarali, *Technology adoption and emerging trends*, Oct. 8, 2021. DOI: 10.1002/9781119685265.ch1.

[81] J. Aguilar-Saborit, R. Ramakrishnan, K. Srinivasan, *et al.*, "Polaris: The distributed sql engine in azure synapse," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3204–3216, Sep. 14, 2020. DOI: 10.14778/3415478.3415545.