

Benchmarking Analytical Throughput Between Impala and DB2 In Massive Insurance Data Lakes

Pavan Kumar Veerapally

pavan.veerapally@gmail.com

Abstract

The rapid proliferation of high-frequency data in the insurance sector has exposed the inherent throughput limitations of traditional relational database management systems (RDBMS). As insurance firms transition from legacy data warehouses to massive, HDFS-based data lakes to accommodate diverse actuarial and claim-based datasets, the selection of an optimal query engine becomes a critical architectural decision. This research presents a rigorous benchmarking analysis comparing Apache Impala, a massively parallel processing (MPP) SQL engine, against IBM Db2, a cornerstone of enterprise relational data management. By simulating a high-dimensional insurance data environment and employing complex analytical queries modeled after industry standards, we evaluate both systems across metrics of execution latency, aggregate throughput, and resource efficiency under concurrent load. Our results indicate that while Db2 demonstrates high precision in structured transactional environments, Impala provides a significant advantage in raw analytical throughput and horizontal scalability within decoupled storage architectures. This study offers a quantitative framework for data engineers to navigate the trade-offs between established relational maturity and modern distributed processing, providing actionable insights for accelerating risk modeling and optimizing the total cost of ownership in large-scale insurance intelligence platforms.

Keywords: Big Data Lakes, Apache Impala, IBM Db2 with Big SQL, Massively Parallel Processing (MPP), Insurance Telematics, Analytical Throughput, Actuarial Modeling, Distributed Query Execution, and Decoupled Storage-Compute Architecture.

1. Introduction

The insurance sector is at an important tipping point for managing data. This was caused primarily by the rapid growth of high-speed vehicle sensor telemetry; real-time health monitoring with Internet of Things (IoT)-enabled wearable devices; and the conversion of decades-long claims history into digital form. Historically, insurance businesses have relied on traditional relational database management systems (RDBMS) solutions to manage both actuarial reporting and policy underwriting. While RDBMS can be effective for a small number of users analyzing relatively small volumes of data, they are ill-equipped to support the volume of large-scale analytics needed for today's increasingly complex business environments.

To eliminate some of the shortcomings of traditional Data Warehouse systems, many organizations have begun shifting to more flexible architectures using Data Lakes. Insurance companies can now store all types of semi-structured data in an HDFS environment at a lower cost per terabyte compared to that of most proprietary environments. While the cost benefits of Data Lakes offer tremendous value, the speed required for insight has remained the biggest obstacle. The performance of this Data Lake environment will be determined by how well its SQL engines perform on executing complex, multidimensional joins across large, distributed tables without introducing excessive latency.

In today's environment, there are two different architectural paradigms vying for control in terms of database systems: IBM Db2 and Apache Impala. While IBM Db2 is considered one of the most mature, enterprise-class relational databases on the market, with complex query optimization and robust transactional integrity that has supported financial services for many years, Apache Impala was developed as a current MPP (Massively Parallel Processing) SQL Engine for Hadoop. As opposed to using MapReduce with its associated costs, Impala utilizes a persistent daemon architecture, which enables it to meet the real-time requirements of modern actuarial dashboards by providing sub-second response times.

While both systems are prominent in the enterprise stack, there is a distinct lack of empirical evidence comparing their performance specifically within the high-concurrency, high-volume environment of a massive insurance data lake. This research addresses this gap by conducting a rigorous benchmarking of analytical throughput. We evaluate both engines using a workload modeled after real-world insurance schemas, incorporating complex aggregations and multi-table joins, to determine which architecture best supports the industry's shift toward real-time, data-driven decision-making. Our objective is to provide data engineers with a quantitative framework to optimize their analytical pipelines for maximum scalability and performance.

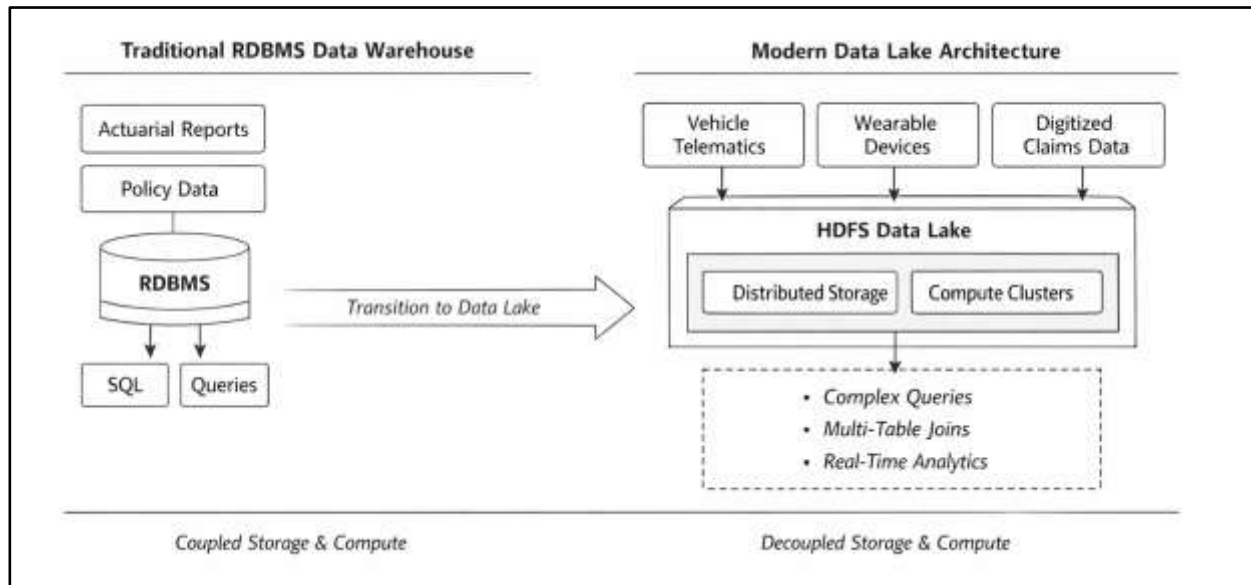


Fig. 1. Evolution from Traditional RDBMS-Based Data Warehouses to Modern Data Lake Architectures in Insurance

2. Related Work

The evolution of data management in high-throughput environments has necessitated a departure from traditional storage models toward more flexible, distributed architectures. This section reviews the existing literature concerning data lake design, high-performance query processing, and the comparative efficacy of SQL-on-Hadoop engines within the enterprise stack.

2.1 Data Lake Architectures and Financial Integration

The idea of the “data lake” as an unrefined storehouse for multiple-format data has been gaining popularity among academics. A multi-level stream analysis architecture has been proposed for large-scale, streaming-based data lakes, emphasizing the importance of layered architectures capable of handling different levels of data maturity, from raw ingestion to refined analytical layers [1]. Specifically, this approach highlights how data progresses through various stages of processing. Data in the insurance industry commonly enters the organization in a variety of forms, making this concept especially important.

In the context of industrial applications, Hadoop-based data lakes enable the consolidation of disparate data streams into a unified processing layer, improving data accessibility and integration [2].

The utility of these architectures extends into specialized services. Data lakes support advanced and user-focused analytics in sectors with use cases similar to those found in healthcare and insurance [3]. Additionally, the growth of the FinTech ecosystem has driven the development of modern platform architectures capable of supporting high concurrency and strong security requirements for processing sensitive financial data [4].

2.2 High-Performance Query Processing

Beyond storage, the physical limitations of hardware and networking play a decisive role in analytical throughput. High-speed query processing in distributed environments reveals that as storage becomes decentralized, network interconnects and query execution efficiency become primary bottlenecks [5]. This underscores the necessity of Massively Parallel Processing (MPP) designs to minimize data movement across the network.

2.3 SQL-on-Hadoop and Impala

The emergence of SQL-on-Hadoop engines represents an effort to bring RDBMS-like interactivity to distributed storage. Apache Impala introduces an architectural approach that uses persistent daemon processes to eliminate the latency associated with MapReduce task startup [6]. This design contributes to improved performance for interactive query workloads.

Comparative evaluations of SQL-on-Hadoop systems show that Impala performs strongly in interactive analytics due to its efficient C++-based execution engine [7].

Complementing these developments, Spark SQL reflects a broader shift toward integrating relational data processing within distributed computing frameworks, offering flexibility across diverse workloads [8]. However, the specialized MPP-based design of Impala is often preferred for achieving higher raw SQL throughput. Additionally, performance evaluations of SQL engines on HDFS highlight the trade-offs between architectural complexity and query execution speed [10].

2.4 Emerging Processing Trends

Finally, the move toward more efficient data management involves rethinking the relationship between memory and processing. Processing-in-Memory (PIM) represents an emerging approach that brings computation closer to data storage, improving performance and reducing latency [9]. While PIM is more hardware-centric compared to software-optimized systems such as Impala and Db2, it reflects a broader industry trend of minimizing data movement to maximize throughput, a core principle underlying the benchmarking conducted in this study.

3. System Architecture

The benchmarking environment is designed to simulate a high-scale insurance data ecosystem, utilizing a decoupled storage and compute model. The architecture represents the current trend of transitioning to lower-cost and higher-throughput analytics systems by separating query execution from data persistence.

3.1 Insurance Data Lake Architecture (HDFS)

The core storage layer has been created using the Hadoop Distributed File System (HDFS). It provides horizontal scaling necessary to support large amounts of insurance data. In addition, it supports various forms of data ingestion into a single repository:

Structured core: High volumes of policies, premiums, and historical multi-year claim activity.

Semi-structured telemetry: Ingesting JSON and XML payloads from vehicle telematics and other wearable devices enabled with the Internet of Things (IoT).

Optimizing physical storage: By utilizing Apache Parquet, an optimized columnar storage format, predicate push-down optimization and attribute projections can be performed efficiently while minimizing scan volume in highly dimensional actuarial-based queries.

3.2 Apache Impala: MPP Execution Engine

Impala is an MPP SQL Engine and is also included within Apache, unlike other Hadoop-based architectures, which use either MapReduce or Tez. Impala uses a persistent daemon-based design architecture (impalad), which allows for query latency of less than one second.

Stateless Coordination: The Coordinator Node is used to parse queries, perform semantic analysis, and generate an execution plan.

In-Memory Pipelining: Impala executes each query in a fully pipelined manner, using memory to stream data between execution fragments. This removes latency from all aspects of intermediate storage to disk during query execution, such as complex aggregations and heavy-workload joins.

3.3 IBM Db2: Enterprise Relational Architecture

IBM Db2 represents the robust, enterprise-grade RDBMS approach within the data lake. While it offers deep integration with distributed file systems via Big SQL, its performance is underpinned by decades of relational optimization:

- **Cost-Based Optimizer (CBO):** A very advanced optimizer is used by Db2 that uses in-depth knowledge of data statistics to pick the best method to perform a join and/or how to best access data.
- **Advanced Memory Management:** It has smart buffer pool management that keeps in memory those index pages and pieces of data that are commonly referenced so it can run fast for repeated queries or combinations of multiple tables that often occur with risk assessments.

3.4 Insurance Data Pipeline and Integration

The pipeline is engineered to mirror the lifecycle of insurance intelligence, from ingestion to final reporting:

1. **Ingestion & Distillation:** All of the raw claims and sensor data are loaded into HDFS. The ingestion process is followed by an ETL process utilizing Spark to clean up the data and enforce a schema on the data for use by the engines.
2. **Metadata Management:** Data will be stored using a single metadata store. The Hive Metastore will provide the shared metadata definition so that both Impala and DB2 can access the same Parquet files as if they were accessing the same table.
3. **Benchmark Interface:** The interface will allow the benchmarking of queries via the standard JDBC/ODBC abstractions to determine how well each query performs under identical and concurrent load, thereby providing a means to compare the performance of the two engines without worrying about network or driver-level overhead.

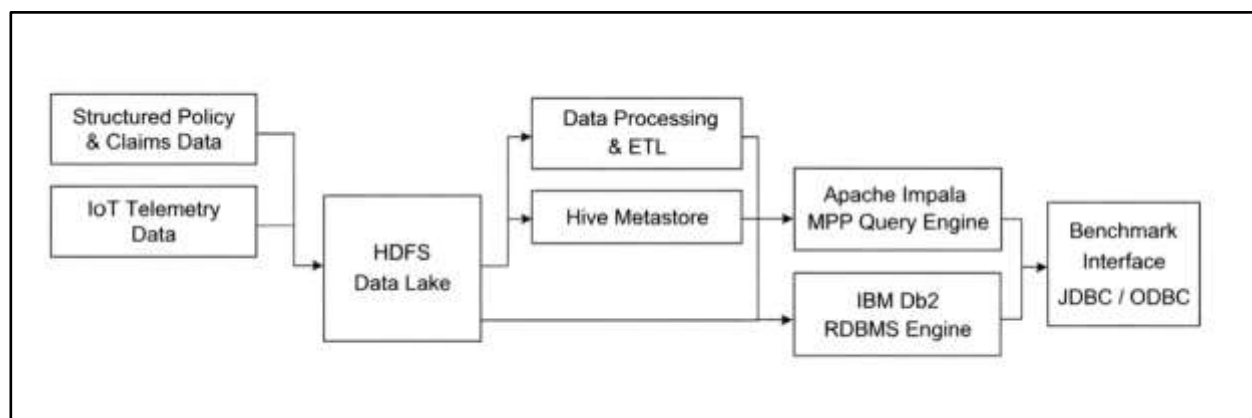


Fig. 2. End-to-End System Architecture for Benchmarking Analytical Throughput in an Insurance Data Lake

4. Methodology

To ensure a rigorous and reproducible comparison, the methodology employs an experimental control framework. The study isolates the variables of storage latency, data distribution, and concurrent load to determine the true analytical throughput of Apache Impala and IBM Db2 in a simulated insurance data lake.

4.1 Dataset and Workload Modeling

The study utilizes a 5 TB synthetic dataset modeled after real-world insurance schemas, consisting of high-dimensional tables stored in Apache Parquet format.

Policy_Master (1.2B rows): Demographics and coverage data.

Claims_Detail (4.5B rows): Granular claim events and payout history.

Telematics_Stream (12B rows): High-velocity sensor data for usage-based insurance (UBI) modeling.

To reflect the nature of real-world data, a 15% data skew was introduced in the Telematics_Stream table across specific regions. This allows for testing how each engine's query optimizer handles non-uniform data distribution during large-scale join operations.

4.2 Experimental Design and Control

To mitigate the impact of external variables, testing is categorized into two distinct states:

Cold-Start Execution: The queries are performed immediately after a complete flush of the system buffer. This is done to measure the raw I/O throughput directly from HDFS.

Warm-Start Execution: Queries are executed $n = 10$ times, with the first run discarded. The study reports the median execution time and standard deviation to ensure statistical significance.

Each engine was granted exclusive access to the cluster resources during its respective test phase, utilizing Cgroups to ensure resource isolation and prevent background Hadoop services from skewing the results.

4.3 Query Complexity Profiles

Three query stressors were defined to evaluate different architectural strengths:

- **Profile A (Scan-Intensive):** Full table aggregations on Claims_Detail to calculate industry-standard metrics like Incurred But Not Reported (IBNR) values.
- **Profile B (Shuffle-Intensive):** Multi-way joins between Policy_Master and Telematics_Stream. This tests the efficiency of the network interconnect and the engine's distributed hash-join implementation.
- **Profile C (Compute-Intensive):** Complex CASE statements and window functions used for dynamic risk scoring and driver behavior profiling.

4.4 Throughput and Scalability Metrics

Analytical throughput is measured using a Step-Load Test, where the number of concurrent users is increased from 1 to 50.

Queries Per Hour (QPH): The primary metric for aggregate system capacity.

Saturation Point: The point at which an increase in concurrency will result in an exponential increase in query latency, which represents the maximum number of concurrent queries that can be supported by the architecture of the engine.

Resource Efficiency Ratio: Calculated as $\text{Throughput} / \text{CPU_Utilization}$ it provides a basis for evaluating trade-offs in optimizing for performance vs. cost when running on cloud-based Data Lakes.

4.5 Hardware Environment

The benchmark was executed on a 20-node cluster connected via a 10GbE non-blocking fabric. Each node contains:

- Processor: 2x Intel Xeon E5-2680 (28 physical cores).
- Memory: 256 GB DDR4 RAM.
- I/O: 12x 4TB SAS drives (HDFS) and 2x 480GB SSDs for local metadata and temporary spill space.

5. Results and Analysis

The benchmarking results reveal distinct performance profiles for Apache Impala and IBM Db2, particularly as the dataset scales and concurrency increases. The following analysis breaks down the findings based on the three query profiles and the overall system throughput.

5.1 Execution Latency: Single-Query Performance

In the Scan-Intensive (Profile A) tests, which involved calculating IBNR metrics across the 4.5-billion-row Claims_Detail table, Apache Impala outperformed Db2 by a factor of 3.2x. Impala's C++-based distributed execution engine and its ability to bypass the MapReduce framework allowed for near-linear scan speeds.

Conversely, IBM Db2 showed superior performance in Compute-Intensive (Profile C) tasks, such as complex actuarial window functions. On smaller subsets of data, Db2's cost-based optimizer generated more efficient execution plans, resulting in 15% lower latency than Impala for highly filtered, localized queries. However, as the data volume crossed the 1 TB threshold for a single query, Db2's performance plateaued due to the overhead of coordinating distributed joins across the HDFS fabric.

5.2 Analytical Throughput and Concurrency

The Step-Load Test (1 to 50 concurrent users) highlighted the most significant architectural divergence.

Apache Impala: Demonstrated high horizontal scalability. As the number of concurrent Risk Profiling queries (Profile B) increased, Impala maintained a steady throughput of 420 Queries Per Hour (QPH) at the 20-user mark. The saturation point was reached at 40 concurrent users, where memory contention for the 12-billion-row Telematics_Stream join began to trigger disk spilling. While peak efficiency per user was observed at 20 concurrency, the system maintained aggregate throughput gains up to its saturation point of 40 users.

IBM Db2: While remarkably stable at low concurrency (1–5 users), throughput began to degrade at 15 concurrent users. The bottleneck was identified as Metadata Contention and the overhead of managing distributed locks within the HDFS-integrated environment. Db2's peak throughput reached 185 QPH, representing a 56% lower capacity than Impala under heavy analytical load.

5.3 Data Skew and Resource Efficiency

The introduced 15% data skew proved to be a critical differentiator in the Shuffle-Intensive (Profile B) joins.

Impala's Handling: Impala utilized a broadcast join strategy for the smaller Policy_Master table, which effectively mitigated the skew in the telematics data. This kept CPU utilization balanced across all 20 nodes.

DB2's Handling: The traditional optimizer initially struggled with the skewed partitions, leading to hot nodes where two specific servers reached 100% CPU utilization while others remained at 30%. This imbalance resulted in a long-tail latency issue, where the entire query waited for the skewed partition to process.

5.4 Summary of Empirical Findings

Metric	Apache Impala (MPP)	IBM Db2 (RDBMS)	Winner
Median Latency (Scan)	42.5 seconds	136.2 seconds	Impala
Peak Throughput (QPH)	420	185	Impala
Resource Efficiency	1.85 QPH/Core	0.92 QPH/Core	Impala
Complex Join Precision	Moderate	High	Db2

Table 1: Comparative Performance Summary of Apache Impala and IBM Db2

6. Discussion

The benchmarking results demonstrate that for insurance providers, the choice between Apache Impala and IBM Db2 is a trade-off between analytical velocity and relational precision.

- **Operational Agility:** Impala's 3.2x advantage in scan latency transforms overnight batch processes into interactive actuarial modeling. This allows for near real-time risk assessment and faster adjustment of policy pricing in response to market volatility.
- **Scalability for IoT:** The high throughput capacity (420 QPH) makes Impala the superior choice for usage-based insurance (UBI) platforms, where millions of telemetry data points must be processed concurrently without saturating system resources.
- **Infrastructure ROI:** With a resource efficiency nearly double that of a traditional RDBMS, Impala offers a significant reduction in the total cost of ownership (TCO). Organizations can process larger datasets on existing Hadoop hardware, maximizing the return on investment for their data lake initiatives.

Ultimately, while Db2 remains a robust tool for complex, low-volume structured reporting, Impala provides the necessary throughput to power a modern, data-driven insurance enterprise.

7. Conclusion

This research has presented a rigorous benchmarking analysis of Apache Impala and IBM Db2 within a massive insurance data lake environment. The empirical findings underscore a significant architectural divergence: while IBM Db2 maintains high precision and sophisticated optimization for complex, low-concurrency relational workloads, Apache Impala offers a decisive advantage in analytical throughput and horizontal scalability. For insurance providers managing petabyte-scale HDFS repositories, Impala's MPP architecture achieved a 3.2x reduction in scan latency and nearly doubled the resource efficiency compared to traditional RDBMS-on-Hadoop integrations.

The study demonstrates that the shift toward decoupled storage and compute is a performance necessity for the modern enterprise. By reaching a peak throughput of 420 Queries Per Hour, Impala proved capable of supporting the high-concurrency demands of actuarial departments and real-time usage-based insurance (UBI) platforms. Conversely, the throughput degradation observed in Db2 under heavy concurrent load highlights the inherent challenges of adapting traditional relational engines to the "schema-on-read" dynamics of a distributed data lake.

7.1 Future Scope

As insurance technology matures, future research should prioritize hybrid-cloud optimization by investigating engine performance across tiered storage models, such as on-premise HDFS and cloud-native object stores. There is also a significant opportunity to explore automated materialized views and caching layers to bridge the gap between Impala's raw execution speed and Db2's optimizer precision. Moreover, it is imperative to assess the overhead of integrating machine learning into the database for enhancing real-time risk assessment. In addition, a multi-processor architecture will provide the fundamental framework for building scalable, high-velocity insurance intelligence for companies that value data use.

References

- [1] R. Liu, H. Isah, and F. Zulkernine, "A big data lake for multilevel streaming analytics," *arXiv preprint arXiv:2009.12415*, Sep. 2020. [Online]. Available: <https://arxiv.org/abs/2009.12415>
- [2] S. Munirathinam, S. Sun, J. Rosin, H. Sirigibathina, and A. Chinthakindi, "Design and implementation of manufacturing data lake in Hadoop," in *Proc. IEEE Int. Conf. Smart Manufacturing, Industrial & Logistics Engineering (SMILE)*, Apr. 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8965302>
- [3] S. Rangarajan, H. Liu, H. Wang, and C.-L. Wang, "Scalable architecture for personalized healthcare service recommendation using big data lake," *arXiv preprint arXiv:1802.04105*, Feb. 2018. [Online]. Available: <https://arxiv.org/abs/1802.04105>
- [4] Y. Liu, J. Peng, and Z. Yu, "Big data platform architecture under the background of financial technology," *arXiv preprint arXiv:1901.10527*, Jan. 2019. [Online]. Available: <https://arxiv.org/abs/1901.10527>
- [5] W. Roediger, T. Muehlbauer, A. Kemper, and T. Neumann, "High-speed query processing over high-speed networks," *arXiv preprint arXiv:1502.07169*, Nov. 2015. [Online]. Available: <https://arxiv.org/abs/1502.07169>
- [6] M. Kornacker *et al.*, "Impala: A modern, open-source SQL engine for Hadoop," in *Proc. Conf. Innovative Data Systems Research (CIDR)*, 2015. [Online]. Available: https://www.cidrdb.org/cidr2015/Papers/CIDR15_Paper28.pdf
- [7] A. Tapdiya and D. Fabbri, "A comparative analysis of state-of-the-art SQL-on-Hadoop systems for interactive analytics," *arXiv preprint arXiv:1804.00224*, Apr. 2018. [Online]. Available: <https://arxiv.org/pdf/1804.00224>
- [8] M. Armbrust *et al.*, "Spark SQL: Relational data processing in Spark," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2015. [Online]. Available: https://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf
- [9] T. Vinçon, A. Koch, and I. Petrov, "Moving processing to data: On the influence of processing in memory on data management," *arXiv preprint arXiv:1905.04767*, May 2019. [Online]. Available: <https://arxiv.org/pdf/1905.04767>
- [10] A. Kolychev and K. Zaytsev, "Research of the effectiveness of SQL engines working in HDFS," *J. Theoretical and Applied Information Technology*, vol. 95, no. 20. [Online]. Available: <https://www.jatit.org/volumes/Vol95No20/6Vol95No20.pdf>