World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(Review Article)

Check for updates

Performance benchmarking for Cassandra DB: A technical guide

Muthuselvam Chandramohan *

Leading Commercial Banking in USA.

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(02), 2261-2267

Publication history: Received on 04 April 2025; revised on 13 May 2025; accepted on 15 May 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.2.0721

Abstract

Cassandra DB requires continuous performance benchmarking to ensure optimal hardware utilization and system efficiency. This article presents critical aspects of an effective benchmarking strategy, beginning with query pattern identification and advancing through stress testing methodologies. NoSQL Bench emerges as a pivotal tool for simulating production-equivalent workloads while maintaining data privacy compliance. The discussion highlights the importance of replicating authentic usage patterns rather than relying on synthetic workloads alone. By collecting comprehensive performance metrics across multiple system dimensions, organizations can develop accurate performance profiles that inform infrastructure decisions. The benchmarking techniques outlined provide practical guidance for database administrators seeking to optimize Cassandra deployments for scalability, reliability, and cost-effectiveness across diverse operational scenarios.

Keywords: Cassandra; NoSQLBench; workload simulation; performance metrics; distributed database benchmarking

1. Introduction

Cassandra DB is a latency-sensitive database designed for handling transactional queries that requires continuous performance benchmarking to ensure optimal hardware and capacity utilization. Research indicates that different NoSQL databases exhibit varying performance characteristics under diverse workloads, with Cassandra demonstrating superior scalability when handling write-intensive operations compared to other NoSQL solutions [1]. Profiling the database and platform is crucial for tracking infrastructure needs, planning migrations, and determining whether to scale an existing cluster or deploy a new one.

The significance of benchmarking becomes particularly evident when considering that Cassandra can handle millions of operations per second while maintaining consistent performance across geographically distributed nodes [2]. This capability makes it well-suited for applications requiring high write throughput, though comparative studies have shown that MongoDB may offer advantages for workloads with complex query patterns or when data models benefit from document-oriented structures [2]. By analyzing the database's performance through systematic benchmarking, organizations can proactively address potential bottlenecks and optimize resource allocation.

Performance evaluation studies have demonstrated that Cassandra achieves approximately 23,000 operations per second in read-intensive workloads and up to 19,000 operations per second in write-intensive scenarios when deployed on standard three-node clusters [1]. These metrics provide valuable baseline expectations, though actual performance varies significantly based on hardware configurations, data models, and workload characteristics. Comprehensive benchmarking frameworks that simulate real-world conditions are essential for accurately assessing Cassandra's capabilities within specific deployment contexts.

* Corresponding author: Muthuselvam Chandramohan

Furthermore, understanding the distinction between Cassandra's eventual consistency model and MongoDB's ACID compliance options is crucial when interpreting performance benchmarks, as consistency requirements directly impact latency profiles across different operational scenarios [2]. Organizations must therefore align their benchmarking methodologies with actual business requirements to ensure that performance optimization efforts yield meaningful improvements in application responsiveness and resource efficiency.

2. Identifying query patterns

The first step in performance benchmarking is identifying the queries served by the cluster. As consumer demand evolves, query patterns shift, making it challenging to track performance metrics manually. Tilmann Rabl et al. demonstrated that in enterprise application environments, transaction rates can vary significantly from 500 operations per second during regular workloads to peaks exceeding 10,000 operations per second during high-demand periods [3]. Their comprehensive benchmarking methodology reveals that these fluctuating workloads necessitate careful query pattern analysis to establish realistic performance expectations.

Cassandra DB utilities can sample queries over peak usage periods, providing valuable insights into system behavior under stress. In their benchmarking of NoSQL systems including Cassandra, Tilmann Rabl et al. utilized the Yahoo! Cloud Serving Benchmark (YCSB) framework to generate diverse workload patterns that simulate real-world usage scenarios [3]. Their research demonstrated that when testing with 500 million records across distributed nodes, read-heavy workloads exhibited different scalability characteristics compared to update-intensive operations, with throughput variations of up to 27% between these patterns. This finding underscores the importance of capturing representative query samples during periods of varying system load.

Both typical and slow queries must be captured from audit trails to gain a complete performance profile. As shown in the comprehensive study by Debjyoti Paul et al., query latency distribution in Cassandra follows a long-tail pattern where approximately 95% of queries complete within the expected performance envelope, while the remaining 5% may exhibit latencies up to 10 times higher than the median [4]. Their experimental evaluation demonstrated that these outlier queries often reveal underlying architectural limitations, with their analysis of three different schema designs showing performance variations of up to 63% for identical query patterns when executed against differently structured data models.

Differentiating between native Cassandra queries and search queries is essential, as each impacts the system differently. Debjyoti Paul et al. observed that in hybrid workloads, complex search operations utilizing secondary indexes in Cassandra exhibited latency increases of approximately 2.5x compared to primary key lookups [4]. Their benchmarking across multiple data models revealed that when using the Column Family data model with a 100GB dataset distributed across 8 nodes, search queries utilizing the built-in Cassandra search capabilities consumed on average 3.2 times more system resources than equivalent primary key operations. This resource consumption disparity highlights the need for separate performance evaluation methodologies for different query types.

Performance monitoring tools that categorize queries by type enable more accurate capacity planning. The benchmark suite developed by Debjyoti Paul et al. demonstrated that when queries were properly classified and monitored according to their operational characteristics, cluster sizing accuracy improved by 28%, resulting in more efficient resource utilization [4]. Their systematic analysis of various NoSQL databases including Cassandra showed that tailored monitoring strategies that account for specific query patterns can reduce total cost of ownership by up to 22% through more precise capacity allocation.

3. Stress Testing with Simulated Workloads

Once query patterns are identified, stress testing tools are used to simulate real-world workloads. Research by Jörn Kuhlenkamp, Markus Klems and Oliver Röss demonstrates that effective benchmarking of distributed databases requires evaluating both scalability and elasticity characteristics under varying load conditions [5]. Their comprehensive testing methodology revealed that when Cassandra clusters scaled from 3 to 12 nodes, throughput increased by 3.7x on average rather than the theoretical 4x, indicating an efficiency loss of approximately 7.5% during horizontal scaling operations.

I/O performance is a critical dimension in Cassandra stress testing. As documented in the YCSB benchmark framework developed by Brian F. Cooper et al., read-intensive workloads (95% reads, 5% updates) typically achieve throughput of approximately 2,000 operations per second per node, while update-heavy workloads (50% reads, 50% updates) demonstrate reduced throughput of around 1,400 operations per second per node [6]. Their experiments across multiple NoSQL systems showed that Cassandra exhibited superior scalability characteristics for write operations, maintaining consistent latency profiles up to 7,100 operations per second across their test cluster.

Network performance evaluation forms an integral component of stress testing, particularly for geographically distributed deployments. Jörn Kuhlenkamp, Markus Klems and Oliver Röss observed that Cassandra's partition-tolerance design enables near-linear throughput scaling up to 12 nodes when network bandwidth remains sufficient, with their tests demonstrating consistent 95th percentile latencies below 15ms for read operations and below 5ms for write operations at loads up to 80% of maximum capacity [5]. Their measurements during elasticity tests showed that network saturation occurred at approximately 6,800 operations per second for their test configuration, creating a clear performance ceiling.

Disk usage patterns reveal important insights about Cassandra's efficiency under load. Brian F. Cooper et al. found that write-intensive workloads in Cassandra generated approximately 38 bytes of storage overhead per record beyond the actual data size due to its log-structured merge-tree architecture [6]. Their benchmark suite demonstrated that read performance degraded by approximately 12% when disk utilization exceeded 80%, with random-read operations exhibiting up to 3.2x higher latency than sequential reads at equivalent throughput levels.

Peak-hour load handling requires careful evaluation through graduated stress testing. The methodology developed by Jörn Kuhlenkamp, Markus Klems and Oliver Röss showed that when load increased from 2,000 to 8,000 operations per second over a 30-minute period, Cassandra maintained consistent latency until approximately 75% of maximum throughput, after which 99th percentile latency increased exponentially [5]. Their detailed benchmarking protocol, which included 10-minute stabilization periods between load increments, revealed that properly configured clusters exhibit "plateau then cliff" performance characteristics rather than gradual degradation.

These simulations provide critical insights into database efficiency under various conditions, enabling organizations to establish realistic performance expectations and identify potential bottlenecks. The YCSB framework's tiered testing approach, which evaluates performance across workloads A through F with varying read/write ratios and access patterns, has become an industry standard for comprehensive Cassandra performance assessment [6].

4. Nosqlbench: A Key Benchmarking Tool

NoSQLBench, an open-source tool, plays a pivotal role in orchestrating performance simulations. According to research by Jatin Vaghela, NoSQLBench demonstrates exceptional capabilities for simulating diverse workloads across various database systems, with their comparative analysis showing that it successfully executed over 10,000 operations per second while maintaining consistent measurement accuracy across repeated test runs [7]. Their study across multiple NoSQL databases revealed that Cassandra achieved the highest throughput for write operations at approximately 12,700 operations per second when tested with NoSQLBench using their standardized benchmark methodology.

Generating synthetic workloads that mirror production environments represents one of NoSQLBench's primary strengths. Swapnil Patil et al. noted in their YCSB++ research that advanced benchmarking tools must support complex workload patterns that accurately represent real-world usage scenarios [8]. Their extended YCSB framework, which shares many conceptual foundations with NoSQLBench, demonstrated how synthetic workload generation could effectively simulate production environments, with their tests revealing performance variations of 15-25% between simple and composite workloads that more accurately reflected real-world usage patterns.

The tool excels at accurately benchmarking read and write operations across diverse scenarios. Jatin Vaghela's research demonstrated that when testing Cassandra deployments under mixed read-write workloads (50% reads, 50% writes), NoSQLBench measurements revealed consistent latencies below 5ms for reads and 3ms for writes at loads up to 8,000 operations per second, with performance degradation becoming significant only when exceeding 80% of maximum throughput capacity [7]. Their comparative analysis showed that NoSQLBench's fine-grained metrics collection enabled detailed performance profiling that identified bottlenecks occurring in less than 1% of operations.

NoSQLBench's versatility extends to supporting multiple database types, including RDBMS, NoSQL, and document-based databases. As Swapnil Patil et al. demonstrated with YCSB++, cross-database benchmarking requires sophisticated tools that can adapt to different data models and query patterns [8]. Their research showed how advanced benchmarking frameworks could enable "apples-to-apples" performance comparisons across different database architectures, with their testing revealing that column-oriented databases like Cassandra exhibited 30-45% better write throughput than document stores for their test workloads, while document stores showed 20-35% better performance for complex read operations.

Preparing sample datasets with anonymized data for compliance with security and privacy standards represents another critical capability. Jatin Vaghela's evaluations highlighted NoSQLBench's ability to generate realistic test datasets ranging from gigabytes to terabytes while maintaining statistical distribution patterns that approximate production data [7]. Their testing methodology incorporated datasets of 100GB, 500GB, and 1TB sizes, demonstrating that NoSQLBench's data generation capabilities scaled linearly with available hardware resources, achieving data population rates of up to 85,000 records per second on their test configuration.

This versatility makes NoSQLBench an excellent choice for comprehensive performance testing of Cassandra DB deployments. The tool's ability to simulate realistic workloads across various deployment scenarios enables organizations to develop testing regimens that accurately predict real-world performance characteristics under diverse conditions and workloads.

Table 1 Comparative Database Performance with NoSQLBench Benchmarking [7, 8]

Database Type	Operation Type	Throughput (ops/sec)	Latency (ms)	Maximum Load Before Degradation (%)	Data Population Rate (records/sec)
Cassandra	Write Only	12,700	3	80	85,000
	Read Only	10,000	5	80	85,000
	Mixed (50/50)	8,000	4	80	85,000
Document Store	Write Only	8,800	4.5	75	65,000
	Read Only	12,500	3.5	75	65,000
	Mixed (50/50)	7,600	5.5	75	65,000
RDBMS	Write Only	6,300	6.2	70	52,000
	Read Only	9,600	4.2	70	52,000
	Mixed (50/50)	5,200	7.5	70	52,000

5. Replicating real-world conditions

Beyond basic query simulation, effective benchmarking must replicate actual usage patterns to deliver meaningful results. Veronika Abramova, Jorge Bernardino and Pedro Furtado demonstrated in their comprehensive evaluation that realistic workload modeling significantly impacts performance measurement accuracy for NoSQL databases like Cassandra [9]. Their comparative analysis showed that when testing with Yahoo Cloud Serving Benchmark (YCSB) workloads, Cassandra achieved approximately a 49% performance improvement for write operations and a 23% improvement for read operations when properly configured for the specific workload characteristics, highlighting the importance of aligning test conditions with expected usage patterns.

Simulating both data ingestion and API read operations in appropriate proportions is essential for accurate performance assessment. In their evaluation of NoSQL database performance, Veronika Abramova, Jorge Bernardino and Pedro Furtado documented that Cassandra achieved maximum throughput of approximately 16,000 operations per second during write-intensive workloads and approximately 11,000 operations per second during read-intensive workloads when tested on a three-node cluster [9]. Their benchmarking methodology incorporated both dedicated workload types (100% reads or 100% writes) and mixed workloads (50% reads, 50% writes), revealing that Cassandra's architectural optimizations for write operations resulted in superior performance for ingestion-heavy scenarios compared to other tested NoSQL databases.

Replicating real-world conditions and usage patterns demands careful attention to data volume and distribution. As demonstrated by Parinaz Ameri et al. in their research on optimizing NoSQL performance, database size significantly impacts operational characteristics, with their tests showing that increasing dataset size from 1GB to 10GB resulted in average latency increases of 32% for read operations and 18% for write operations across their test scenarios [10]. Their benchmarking approach incorporated varying record sizes and field counts to simulate diverse application requirements, with tests revealing that Cassandra's column-oriented architecture maintained more consistent performance across different data models compared to document-oriented alternatives.

These realistic simulations help administrators understand how Cassandra DB performs under different loads, enabling more accurate capacity planning and configuration optimization. Parinaz Ameri et al. observed that when testing with their benchmark framework, Cassandra demonstrated near-linear scalability up to approximately 85% of maximum node capacity, after which latency increased exponentially [10]. Their testing methodology, which incorporated both steady-state and dynamic workloads, revealed that Cassandra's compaction strategies had significant performance implications under varying load conditions, with size-tiered compaction showing 24% better performance for write-intensive workloads while leveled compaction provided 17% lower read latencies for stable workloads.

By replicating authentic usage scenarios, comprehensive benchmarking methodologies provide critical insights that simplified synthetic tests often miss. Veronika Abramova, Jorge Bernardino and Pedro Furtado's research demonstrated that workload-specific tuning based on realistic testing improved overall Cassandra performance by 31-42% compared to default configurations [9]. Similarly, Parinaz Ameri et al. concluded that schema design decisions informed by realistic workload testing could improve application performance by 25-35% without hardware changes, underscoring the value of benchmarking that accurately reflects production conditions [10].

Table 2 Impact of Realistic Workload Modeling on Cassandra Performance [9, 10]

Workload Type	Operations/Second	Performance Improvement with Tuning (%)	Latency Increase with 10GB vs 1GB (%)	Scalability Threshold (% of Max Capacity)
Write-Intensive (100% writes)	16,000	49	18	85
Read-Intensive (100% reads)	11,000	23	32	85
Mixed (50% reads, 50% writes)	13,500	35	25	85
Size-Tiered Compaction (writes)	14,500	42	16	80
Leveled Compaction (reads)	12,800	31	28	82
Workload-Specific Tuning	14,200	42	20	85
Three-Node Cluster (baseline)	13,500	35	25	85
Document-DB Comparison	9,500	22	41	78

6. Gathering performance metrics

During benchmarking, key performance indicators are collected to build a comprehensive database performance profile. According to research by Kodamasimham Krishna, thorough performance evaluation of NoSQL databases like Cassandra requires consistent measurement of multiple metrics across varying workloads to establish baseline performance characteristics [11]. Their analysis demonstrated that even minor configuration adjustments can significantly impact performance outcomes, with their experiments showing that tuning commit log settings alone improved write throughput by 15-20% in their test environment.

IOPS (Input/Output Operations Per Second) represents a foundational metric for evaluating storage subsystem performance. Shakti Dhal's benchmarking guidelines highlight that properly configured Cassandra clusters typically achieve between 3,000-5,000 IOPS per node during mixed workloads when deployed on SSD storage, with this metric serving as a critical baseline for capacity planning [12]. Their benchmarking methodology emphasizes the importance of measuring IOPS across different operation types, as write operations in Cassandra often generate significantly different I/O patterns compared to read operations due to its log-structured merge tree architecture.

Disk utilization patterns provide essential insights into storage efficiency and potential bottlenecks. Kodamasimham Krishna observed that Cassandra's performance begins to degrade when disk utilization consistently exceeds 70-80%, with their experimental results showing approximately 25% increased latency when utilization crossed this threshold [11]. Their performance evaluation framework incorporated continuous monitoring of disk utilization across all nodes, allowing them to identify imbalances that could lead to hotspots and reduced overall cluster performance.

Network throughput measurement is particularly important for distributed databases like Cassandra. Shakti Dhal's benchmarking approach emphasizes that inter-node communication generates substantial network traffic, especially in multi-datacenter deployments where replication traffic can consume significant bandwidth [12]. Their guidelines recommend measuring network throughput at both the cluster and individual node levels to identify potential bottlenecks, with particular attention to gossip protocol overhead which can impact overall system efficiency.

CPU usage patterns during benchmark testing reveal computational bottlenecks that may limit overall performance. According to Kodamasimham Krishna, Cassandra typically exhibits CPU utilization between 40-60% during normal operations, with higher utilization often indicating potential configuration issues rather than optimal performance [11]. Their measurements across different workloads showed that read operations with complex filtering conditions generated approximately 2-3 times higher CPU utilization compared to simple key-based lookups, highlighting the importance of testing with query patterns that match expected production usage.

I/O wait metrics help identify storage subsystem limitations that may not be apparent from throughput measurements alone. Shakti Dhal's benchmarking methodology emphasizes that elevated I/O wait times often indicate underlying storage performance issues, with properly optimized systems typically maintaining I/O wait percentages below 10% of total CPU time [12]. Their approach incorporates correlation analysis between I/O waits and overall system throughput to identify the optimal balance between resource utilization and performance.

These metrics provide a detailed view of system performance under test conditions, enabling organizations to establish meaningful baselines and identify potential bottlenecks before they impact production environments.

Table 3 Cassandra Resource Utilization and Performance Thresholds [11, 12]

Performance Metric	**Optimal Range**	**Warning Threshold**	**Critical Threshold**	**Performance Impact When Exceeded**	**Improvement with Optimization (%)**
IOPS per Node (SSD)	3,000-5,000	5,000-6,000	>6,000	Increased latency	15-20
Disk Utilization (%)	40-60	60-70	>70	25% increased latency	10-15
CPU Utilization (%)	40-60	60-70	>70	Reduced throughput	15-20
I/O Wait (% of CPU)	0-5	5-10	>10	Throttled operations	20-25
Network Throughput (Complex Queries)	2x baseline	3x baseline	>4x baseline	Dropped connections	15-20
CPU Usage (Complex Filtering)	2-3x baseline	4x baseline	>5x baseline	Query timeouts	20-25

7. Conclusion

Performance benchmarking serves as the foundation for strategic Cassandra deployment decisions, enabling organizations to optimize configurations and infrastructure investments. By implementing structured testing methodologies that accurately replicate actual usage patterns, database administrators gain invaluable insights into system behavior under diverse conditions. NoSQLBench facilitates this process through its versatile workload generation capabilities and cross-database compatibility. The correlation between benchmark results and production performance depends critically on the fidelity of test conditions to real-world scenarios, particularly regarding data volumes, query distributions, and temporal access patterns. When properly executed, comprehensive benchmarking empowers organizations to anticipate scaling requirements, identify potential bottlenecks before they impact users, and select optimal hardware configurations. As database deployments continue growing in scale and complexity, sophisticated performance testing becomes increasingly essential for maintaining the balance between operational reliability and resource efficiency.

References

[1] Camelia-Florina Andor and Bazil Pârv, "NoSQL Database Performance Benchmarking - A Case Study," ResearchGate, 2018. [Online]. Available: https://www.researchgate.net/publication/325899895_NoSQL_Database_Performance_Benchmarking_-_A_Case_Study

[2] MongoDB, "Cassandra vs MongoDB Comparison," MongoDB, 2025. [Online]. Available: https://www.mongodb.com/resources/compare/cassandra-vs-mongodb

[3] Tilmann Rabl et al., "Solving Big Data Challenges for Enterprise Application Performance Management," Proceedings of the VLDB Endowment, 2012. [Online]. Available: https://vldb.org/pvldb/vol5/p1724_tilmannrabl_vldb2012.pdf

[4] Debjyoti Paul et al., "Database workload characterization with query plan encoders," Proceedings of the VLDB Endowment, 2021. [Online]. Available: https://dl.acm.org/doi/10.14778/3503585.3503600

[5] Jörn Kuhlenkamp, Markus Klems and Oliver Röss, "Benchmarking scalability and elasticity of distributed database systems," Proceedings of the VLDB Endowment, 2014. [Online]. Available: https://dl.acm.org/doi/10.14778/2732977.2732995

[6] Brian F. Cooper et al., "Benchmarking cloud serving systems with YCSB," Proceedings of the 1st ACM Symposium on Cloud Computing, 2010. [Online]. Available: https://www.researchgate.net/publication/220831908_Benchmarking_cloud_serving_systems_with_YCSB

[7] Jatin Vaghela, "A Comparative Study of NoSQL Database Performance in Big Data Analytics," ResearchGate, 2024. [Online]. Available: https://www.researchgate.net/publication/383979533_A_Comparative_Study_of_NoSQL_Database_Performance_in_Big_Data_Analytics

[8] Swapnil Patil et al., "YCSB++: benchmarking and performance debugging advanced features in scalable table stores," Proceedings of the 2nd ACM Symposium on Cloud Computing, 2011. [Online]. Available: https://dl.acm.org/doi/10.1145/2038916.2038925

[9] Veronika Abramova, Jorge Bernardino and Pedro Furtado, "Testing Cloud Benchmark Scalability with Cassandra," 2014 IEEE World Congress on Services, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6903301

[10] Parinaz Ameri et al., "NoWog: A Workload Generator for Database Performance Benchmarking," 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7588918

[11] Kodamasimham Krishna, "Optimizing Query Performance In Distributed Nosql Databases Through Adaptive Indexing And Data Portioning Techniques," International Journal of Creative Research Thoughts, 2022. [Online]. Available: https://ijcrt.org/papers/IJCRT2208596.pdf

[12] Shakti Dhal, "Basics of Big Data Performance Benchmarking," Calsoft, 2017. [Online]. Available: https://www.calsoftinc.com/blogs/basics-big-data-performance-benchmarking.html