University of Aden Journal of Natural and Applied Sciences., vol.29(2) (2025) 61-76

University of Aden Journal of Natural and Applied Sciences

Journal homepage: https://uajnas.adenuniv.com

Research Article

Experimental and Functional comparison of BigchainDB and SQL server for Data Management

Maria Othman Saleh Maksha, Khaled Ahmed Abood Omer

Computer Science and Engineering Department, Faculty of Engineering, University of Aden

https://doi.org/10.47372/uajnas.2025.n2.a07

ARTICLE INFO

Received: 05/01/ 2026

Accepted: 19/02/ 2026

Keywords:

Data management, Blockchain, BigchainDB, SQL Server, Performance evaluation, Decentralized databases, functional evaluations

Abstract

The rapid growth of digital data has increased the demand for data management systems that provide not only high performance and scalability but also strong security, integrity, and trust guarantees. This paper presents an experimental and functional comparison between SQL Server, a traditional relational database, and BigchainDB, a Blockchain-based decentralized database that integrates distributed ledger features with database capabilities. Both systems were deployed in an identical containerized simulation environment using Docker to ensure fair and reproducible evaluation. A unified dataset containing up to 100,000 records was generated and used to assess insertion performance, query latency, scalability, and system resource consumption (CPU and memory). System behavior was continuously monitored using Prometheus and Grafana. In addition to performance metrics, functional metrics including immutability, traceability, and ownership control were evaluated.

The experimental results show that SQL Server achieves significantly lower latency and faster query response, but at the cost of higher CPU and memory utilization. Conversely, BigchainDB demonstrates lower resource consumption and provides strong security and tamper-resistance guarantees, though with increased latency due to consensus and transaction validation mechanisms. These findings highlight the trade-offs between centralized and decentralized data management solutions and provide practical guidance for selecting the appropriate technology based on application requirements for performance, trust, and security.

1. Introduction

As a result of the rapid development of data management technology, data is constantly being collected, analyzed, and widely used in various fields, including healthcare, finance, intellectual property management, and government services. Traditional database management systems, particularly relational database management systems (RDBMS), have long served as the primary solution for structured data storage and retrieval due to their high performance, mature query optimization mechanisms, and reliability. They rely on a single administrative authority, which introduces risks such as single points of failure, limited transparency, mistrust, privacy disclosure, and potential data tampering. In applications where trust, auditability, and data integrity are critical, these limitations can affect system reliability and user confidence. Ensuring secure, traceable, and tamper-resistant data management has emerged as a major challenge.

SQL Server is a RDBMS system developed by Microsoft. It provides a robust platform for efficiently storing, retrieving, and managing relational data. SQL Server processes SQL queries for retrieving, manipulating, and managing data. SQL Server optimizes the execution of queries through features such as optimization, indexing, and caching .

Blockchain technology has recently gained attention as a promising alternative for addressing these concerns. By employing decentralization, cryptographic verification, and consensus mechanisms, Blockchain-based systems provide immutable records, transparency, and enhanced trust without relying on a central authority. However, traditional Blockchain platforms are not optimized for high-throughput data storage or efficient querying, making them less suitable for general-purpose database workloads. Their consensus overhead, limited storage capacity, and synchronization costs result in higher latency compared to traditional databases.

* Correspondence to: University of Aden- Yemen
E-mail address: omothman20921@gmail.com
k_abood@hatmail.com

Several studies have shown that ensuring data integrity, validity, security, and privacy has become a new challenge in the development of data management technologies. The rapid growth and widespread adoption of Blockchain technology have led to the need for effective data management strategies in this decentralized and immutable environment. Since Blockchain networks store massive amounts of data across multiple nodes, it is essential to manage and organize this data effectively to ensure the system's integrity, security, and ease of use. Therefore, developing data management is crucial for addressing the unique challenges of data management in Blockchain. Some studies have also highlighted the drawbacks of Blockchain technology, such as limited storage space and slow synchronization times, which make its direct application in the field of big data impractical [1, 2, 3].

Other studies discussed Blockchain technology, its challenges, architecture, and applications in information systems. They explored how Blockchain can be used in data management, highlighting the challenges and how to address them for successful implementation in this field. The advantages of Blockchain have led to its use in data management. This led to the search for a technology that combines the characteristics and advantages of both systems, leading to BigchainDB [4, 5].

To bridge the gap between traditional databases and Blockchain systems, hybrid solutions such as BigchainDB have been proposed. BigchainDB combines Blockchain characteristics, including decentralization, immutability, and asset ownership control, with database features such as high transaction throughput, structured data indexing and querying, low latency, and scalable storage such as MongoDB databases. This design aims to offer a balanced trade-off between performance and trust. MongoDB is considered as one of the NoSQL databases [6] that is used in BigchainDB [5]. Each node in the BigchainDB network contains its own components of MongoDB, a BigchainDB server, and Tendermint. This enables it to store transactions in a decentralized manner, providing transparency and security when tracking ownership rights [7, 8].
Several studies have addressed the efficiency and ease of use of SQL Server, as well as its potential limitations. These studies have demonstrated the use of SQL Server for database management and querying, highlighting its importance in data management. They have explained that SQL Server's structure and underlying architecture are essential for managing and processing data within SQL Server environments. Understanding the components of SQL queries, their syntax, and their usage is crucial for developing, managing, and analyzing databases effectively [9].

Motivated by this gap, this study conducts a comprehensive experimental and functional comparison between BigchainDB and Microsoft SQL Server. The goal is not only to evaluate performance, but also to analyze the trade-offs between centralized and decentralized approaches in terms of execution time, resource consumption, scalability, and functional capabilities such as immutability, traceability, and ownership management. Both systems are deployed under identical conditions within a containerized environment to ensure fairness and reproducibility. Performance metrics are collected using standardized workloads and monitored using Prometheus and Grafana.
The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the research methodology and experimental setup. Section 4 presents and discusses the experimental results. Finally, Section 5 concludes the paper and outlines directions for future work.

2. Related work:

With the rapid development of data management technology and its increased use in various fields, data security issues have increased, including a lack of trust, privacy disclosure, and user privacy violations. Traditional data management approaches, managed and maintained by a single organization, have struggled to solve these problems effectively. At the same time, providing effective guarantees for data authenticity, validity, security, and privacy has become a new challenge in the development of data management technology.

X. Chen et al. proposed an improved data management approach based on Blockchain technology, characterized by decentralization, tamper-proof, and traceability, and addressing the bottlenecks of traditional data management approaches. The study demonstrates the current development of Blockchain technology and describes its characteristics and structure in detail. This study introduced the application of Blockchain technology in data management approaches [10].

J. Chen et al. demonstrated the unique properties of Blockchain technology, such as non-tampering and traceability, that give it significant potential in employee information management and can effectively solve many traditional file management problems. However, it also highlighted the drawbacks of Blockchain technology, such as its limited storage space and slow synchronization time, making it inaccessible to direct application in the field of big data. The study proposed a Blockchain-based personnel management system that built a new model for on-chain and off-chain data storage, effectively solving the problem of data duplication and limited storage space [11].

S. Lupaiescu et al. compared BigchainDB with Amazon QLDB. The study presented a comparative experimental evaluation of the two selected databases, BigchainDB and Amazon Quantum Ledger Database (QLDB). They created three scenarios for both platforms, BigchainDB and QLDB: read (1000,5000,10000) and write (1000,5000,10000). The resource usage of the processor and memory for each one was monitored. The obtained results showed that QLDB performed better than BigchainDB, based on the metrics used. By implementing a ledger database, Amazon QLDB proved to be a comprehensive solution that is easier to use, while BigchainDB involves a more complex implementation and development system, but it is considered more flexible. This flexibility refers to the possibility of self-hosting, customization of network settings, and support for deployment in decentralized public or private environments [12]. This study used up to 10000 records, while in our study, the number of records was increased to 100000 records to address the scalability issue.

A. Alotaibi et al. reviewed current Blockchain data management systems, focusing on two prominent Blockchain databases: BigchainDB and FalconDB. They illustrated architecture, design aspects, and distinctive features of each in detail. The comparison revealed that they share certain characteristics, such as immutability, low latency, authorization, horizontal scalability, decentralization, and the same consensus protocol. However, they differ in terms of database type, synchronization mechanism, replication model, cost, and use of smart contracts [13].

Y. Wang et al. compared the performance of BigchainDB, HadoopDB, and Hive for transaction creation and querying. They showed that BigchainDB outperforms the other systems regarding query execution time and creation execution time. This is because BigchainDB is a decentralized database. HadoopDB is centralized, so writing data needs to go through the center; Hive, which is an offline batch analysis engine, is not suitable for a single record written [14].

A. Rudniy designed and implemented a data warehouse for an online learning platform using three technologies: Microsoft SQL Server, MongoDB, and Apache Hive. The three systems were evaluated in terms of text structure and descriptive analytics. Apache Hive was found to achieve the best processing time, followed by SQL Server and MongoDB. In terms of analytical queries, SQL Server performed best, followed by MongoDB and Hive [15].

C. Ming Wu et al. compared the write efficiency of MS-SQL and MongoDB. In their experiments under indexing conditions, MongoDB was shown to be approximately 10 times faster than MS-SQL in write performance. They found that, under the 1,000,000 operating frequency, the writing performance of MS-SQL or MongoDB is better than their reading performance [16].

A. Malik et al conducted a comparative analysis between Microsoft SQL Server and MongoDB within the context of unstructured JSON data representation. A series of experiments was performed. In the first experiment, 100,000 entries were entered into both SQL Server and MongoDB databases. It was concluded that MongoDB was 15 seconds faster than SQL Server. In the second experiment, a random string was searched 10 times within both databases, and the results of each iteration were compared. The results of these experiments were very intensive, revealing a significant difference between the two databases. In fact, searching for a specific string occurrence was extremely efficient and easy to implement within MongoDB. In the third experiment, the search was performed using a randomly generated identifier. The primary key is the SQL Server ID, while the default index in MongoDB is in the _id field. SQL Server proved to be a fast and efficient tool when searching using the ID field, which contains the primary key [17].

Existing comparative research focuses either on pure Blockchain platforms or on traditional databases, without evaluating both systems under identical deployment conditions. Furthermore, many studies emphasize theoretical analysis rather than practical performance metrics such as execution time, CPU usage, memory consumption, and scalability across varying workloads. As a result, there is a lack of empirical evidence to guide practitioners in selecting appropriate technologies based on both performance and functional trade-offs.

To deal with this gap, the present work provides a systematic experimental and functional comparison between BigchainDB and Microsoft SQL Server using a controlled, containerized environment. Unlike previous studies, our approach evaluates both systems using identical hardware resources, standardized workloads, and continuous monitoring tools. This enables a fair and reproducible assessment of performance characteristics while also examining functional metrics like immutability, traceability, and ownership control.

3. Research Methodology:

This section describes the experimental setup, system configuration, dataset preparation, evaluation metrics, and monitoring framework used to ensure a fair and reproducible comparison between BigchainDB and Microsoft SQL Server.

3.1. Hardware and Software Environment

These experiments are conducted on a Windows 11 machine with an Intel Core i7 processor, 8 GB of RAM, and Docker Desktop (v4.36.1) [18]. The two databases, viz. BigchainDB (v2.2.2) and SQL Server 2019 were used in isolated Docker containers. Also, we used MongoDB 3.6 and Tendermint (v0.31.5) containers for the BigchainDB network (2 nodes). Python 3.9 is used to handle all operations with Prometheus and Grafana monitoring tools in this research work. The same hardware configuration was used for all tests to maintain consistency and comparability.

3.2 Experimental setup:

All experiments were conducted using a containerized simulation environment using Docker Compose to emulate a distributed BigchainDB network on a single machine [19]. Each database system had its own independent container, with CPU and memory resources to ensure consistent resource availability, fair comparison, and minimize the impact of external factors. Identical workloads and dataset sizes were applied to both systems. A containerized simulation for BigchainDB network was also constructed using Docker Compose, with each node hosted in an isolated Docker container with a dedicated MongoDB storage unit. This enabled realistic distributed execution, resource isolation, and reproducible performance measurement. System metrics were collected using Prometheus and visualized with Grafana. Each experiment was repeated multiple times, and average values were recorded to minimize random fluctuations and improve measurement reliability. We ran each system separately to ensure that metric measurements were isolated for fair comparison. Further, the database was reinitialized before each experiment.

3.2.1 BigchainDB Network Architecture

For reliability and to address real-world usage scenarios, the BigchainDB network was built with two nodes, each containing a BigchainDB server, MongoDB, and Tendermint. Node 1 processes transactions as the primary node, while Node 2 receives data and participates in consensus via Tendermint, as shown in Figure 1. All transactions are submitted via a Python script to Node 1, and both nodes were connected to Prometheus and Grafana monitoring tools.

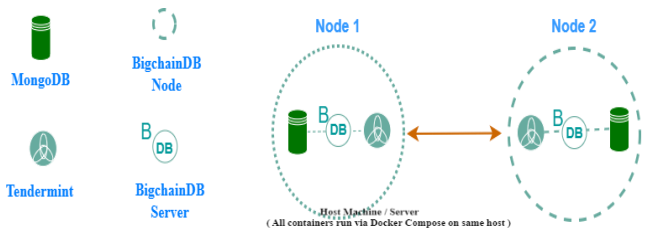


Figure 1. Communications in a Two-Node BigchainDB network

3.2.2 Workflow Diagram:

The following diagram illustrates the workflow for the experimental setup, including the data generation, insertion, query, monitoring, and analysis phases, as shown in Figure 2. The diagram shows the same operations applied to both BigchainDB and SQL Server to ensure fairness. The architecture of the prototype explains interactions among BigchainDB, SQL Server, and the utility services, shown in Figure 3. It illustrates that all containers operate on a single network within Docker, and the BigchainDB network is made of two nodes, as shown. Data is entered into both SQL Server and BigchainDB using the Python script. The interaction between SQL Server and BigchainDB is monitored using the Grafana and Prometheus monitoring tools, which are available on the same Docker network.

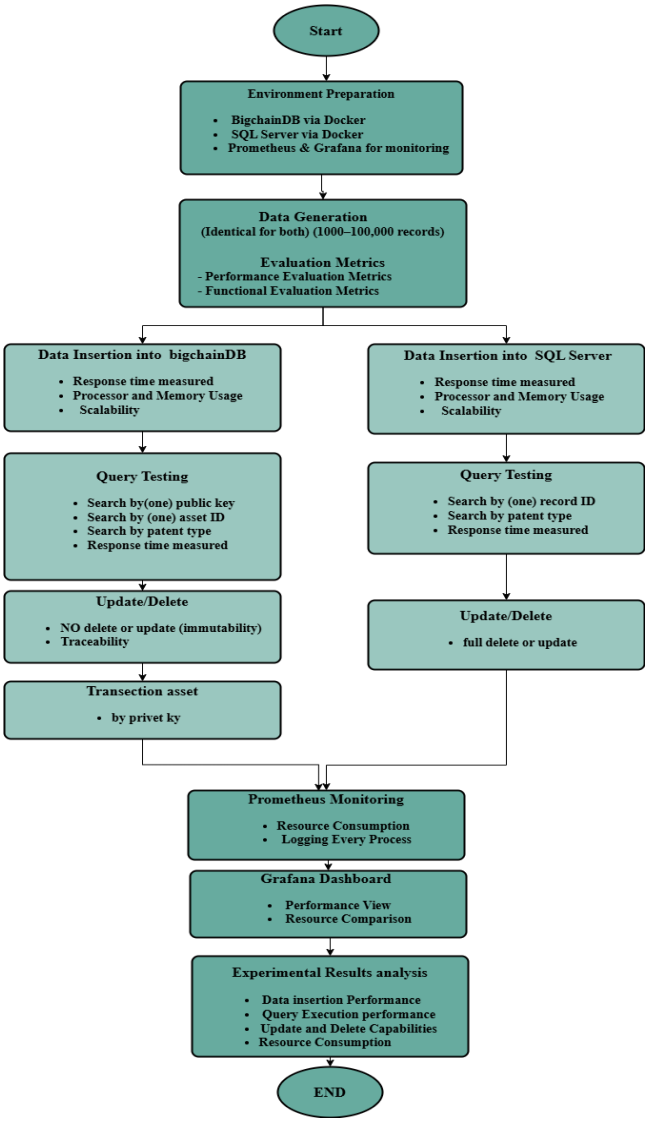


Figure 2 . Flow diagram of the experimental setup

Figure 3 : Architecture of the prototype interacting with BigchainDB and with SQL server

3.3 Dataset Preparation:

To simulate realistic workloads, an intellectual property rights dataset was generated using the Python library faker [20]. Each record of this dataset includes the following fields: recordID/assetID, 'name', 'description', 'owner', 'type', 'status', 'date', 'document_url', and 'public_key' (only for BigchainDB). Dataset sizes were gradually increased to evaluate scalability: 1000, 5000, 10000, 50000, and 100000 records, respectively, to evaluate performance at different scales. The same dataset was used for both systems to ensure identical input conditions.

3.4 System Implementation:

We performed the operations (insert data, query, delete, modify, and measure the time) using Python scripts. For inserting data into SQL Server, we used the **pyodbc** library to execute batch INSERT commands. Similarly, in BigchainDB, we used **bigchaindb-driver** to send CREATE transactions, with each record representing a new "patent."Queries were executed based on Record ID and patent type in SQL Server, or by searching using public key or Asset ID and patent type in BigchainDB. SELECT commands were used in SQL Server, and REST API queries in BigchainDB. Response times for each operation were recorded using precise metrics built into the code. In SQL Server, the UPDATE and DELETE commands were executed directly without any interference. In BigchainDB, due to its immutable nature, data cannot be deleted or modified. However, modifications are represented by sending a new transaction, but all previous transactions remain traceable. These tests highlight differences between mutable relational databases and immutable Blockchain systems.

3.5 Monitoring

Prometheus continuously collects system-level metrics, while Grafana provides real-time visualization. Prometheus was configured within the isolated environment to collect container and system performance metrics using Cadvisor, including CPU and memory consumption for each system. A default data collection period of 15 seconds was used in all experiments. Grafana was used to display and analyze the collected data via a monitoring dashboard. The Grafana dashboard refresh rate was set to 5 seconds to display data during the experiments. The same settings were applied to the systems under investigation to ensure measurement fairness and reproducibility. Monitoring was performed concurrently with workload execution to capture realistic resource behavior under load conditions.

3.6 Evaluation Metrics:

We categorized the evaluation metrics into two categories:

- **Performance Evaluation Metrics**

➢ **Data Entry Time**: The time it takes to enter data into the database of the two systems. Measured in seconds using a Python script.

➢ **Query Execution Time**: The time it takes the system to retrieve a record or set of records based on a specific query.

➢ **Processor and Memory Usage**: This measures CPU and RAM usage during execution. It was monitored using Grafana and Prometheus.

➢ **Scalability**: This measures the system's ability to maintain performance under increased data size or process load.

- **Functional Evaluation Metrics**

➢ **Traceability**: The system's ability to record and track all modifications made to records, ensuring transparency and facilitating auditing and review processes.

➢ **Immutability**: This means that data cannot be changed or deleted once it has been entered and confirmed in the database.

➢ **Ownership Control**: BigchainDB has a concept of owner-controlled assets. Only the owner of an asset can transfer that asset. The owners are the holders of a particular set of private keys. Not even a node operator can transfer an asset [5].

3.7 Fairness Considerations and Limitations

Although both systems were evaluated under identical hardware and workload conditions, architectural differences must be considered for fair comparison. SQL Server follows a centralized design optimized for low-latency queries, whereas BigchainDB employs a decentralized consensus and cryptographic validation, which introduces additional overhead. Therefore, the comparison emphasizes trade-offs between performance and trust guarantees rather than raw speed alone. Additionally, experiments were limited by hardware resources, restricting the maximum dataset size. Future work may extend the evaluation to larger-scale distributed environments.

4. Experimental Results and Discussion

This section presents the experimental results obtained from evaluating BigchainDB and Microsoft SQL Server under

identical deployment conditions. The analysis focuses on insertion performance, query latency, resource utilization, scalability, and functional capabilities.

4.1 Data Insertion Performance

We evaluated the performance of both SQL Server and BigchainDB in processing the generated data. The data was entered in batches ranging from 1000 to 100000 records into both SQL Server and BigchainDB. During each insert, system response time and immediate resource consumption (CPU and memory) were recorded using Python timers and Grafana dashboards, respectively. As shown in Table 1 and Figures 4-23, SQL Server consistently performed better in execution time than BigchainDB, but exhibited significantly higher CPU and memory consumption during insert operations. While monitoring BigchainDB for both nodes, we observed that execution times were large and increased with the number of inserted assets. This is because each inserted asset requires consensus and approval from all nodes before it can be stored. However, they all demonstrated stable resource consumption, operating at similar values. There wasn't a sudden or very large spike compared to SQL Server, which is a very resource-intensive system by nature, but it took a very short time to process, even for operations like inserting 100000 records.

4.1.1 case 1- Insert 1000 records

Table 1 shows the results of inserting 1000 records and assets into both SQL Server and BigchainDB. The table includes the execution time for inserting 1000 records into SQL Server, which is very low compared to inserting the same number of assets into BigchainDB. Figures 4 and 5 show that SQL Server has higher CPU consumption compared to the consumption values in BigchainDB. Figures 6 and 7 show that SQL Server has higher memory consumption compared to the consumption values in BigchainDB.

4.1.2 case 2- Insert 5000

Table 1 shows the results of inserting 5000 records into BigchainDB and SQL Server. The insert execution time is relatively small compared to inserting the same number of assets into BigchainDB. The table also contains CPU and memory consumption values for both systems. Figures 8 and 9 show that SQL Server has higher CPU consumption compared to the consumption values in BigchainDB. Figures 10 and 11 show that SQL Server has higher memory consumption compared to the consumption values in BigchainDB. It is observed that significant resource consumption increases with the increase in the number of records.

Table 1: Results of the insert operation in BigchainDB and SQL server

		BigchainDB				SQL Server	
		Node 1		Node 2			
		Mean	Max	Mean	Max	Mean	Max
Insert 1000	CPU usage	3.54%	4.74%	2.72%	3.67%	5.44%	9.26%
	Memory usage	317MB	325MB	217MB	222MB	856MB	885MB
	Time in second	1254.53S				21.353S	
Insert 5000	CPU usage	4.08%	5.02%	2.21%	2.68%	16.5%	42.9%
	Memory usage	244MB	292MB	162MB	188MB	1.07GB	1.20GB
	Time in second	6355.62S				193.479S	
Insert 10000	CPU usage	4.76%	18.0%	2.28%	4.02%	20.8%	56.0%
	Memory usage	404MB	503MB	230MB	265MB	1.37GB	1.48GB
	Time in second	14166.91S				269.690S	
Insert 50000	CPU usage	4.55%	24.5%	2.31%	5.49%	47.9%	65.1%
	Memory usage	510MB	689MB	252MB	278MB	1.25GB	1.45GB
	Time in second	71846.1S				787.178S	
Insert 100000	CPU usage	4.40%	25.8%	2.29%	5.81%	46.3%	74.8%
	Memory usage	528MB	832MB	226MB	278MB	1.14GB	1.62GB
	Time in second	153905.3S				834.51S	

Figure 4. 1k inserts CPU usage BigchainDB node1&2

Figure 5. 1k inserts CPU usage SQL server

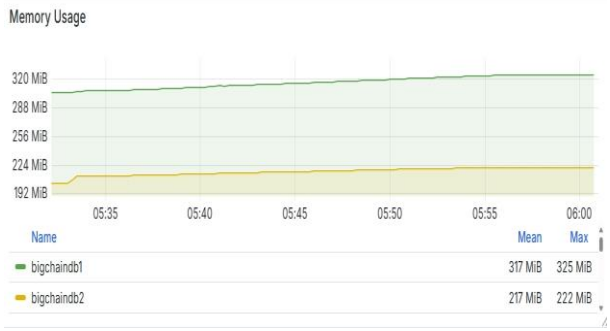


Figure 6 1K inserts memory usage BigchainDB node1&2

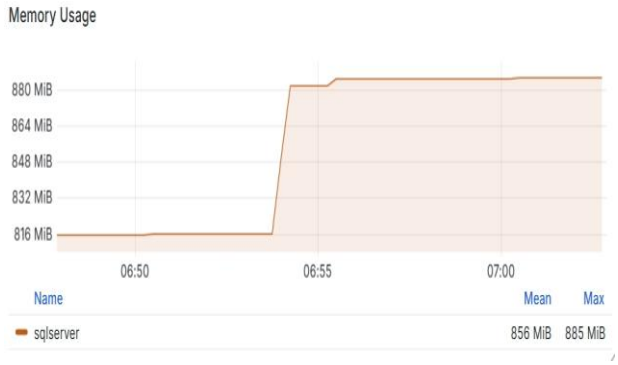


Figure 7 1K inserts memory usage SQL server

Figure 8. 5K inserts CPU usage BigchainDB node1&2

Figure 9. 5K inserts CPU usage SQL server

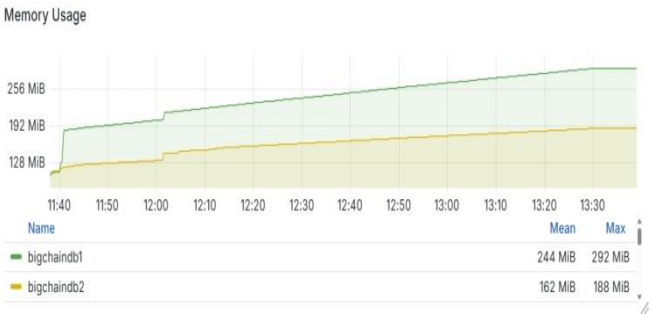


Figure10 5K inserts memory usage BigchainDB node1&2

Figure 11 5K inserts memory usage SQL server

4.1.3 Case 3- Insert 10000

The results of inserting 10000 records into SQL Server are shown in Table 1. The table includes the insert execution time, which is relatively short compared to inserting the same number of assets into BigchainDB. However, the CPU and memory consumption are significantly higher than the previous insert operations, compared to BigchainDB. Figures 12 and 13 show that SQL Server has higher CPU consumption compared to the consumption values in BigchainDB. Figures 14 and 15 show that SQL Server has higher memory consumption compared to the consumption values in BigchainDB. It is observed that significant resource consumption increases with the increase in the number of records.

4.1.4 Case 4 - Insert 50000

As with the previous insertion results, Table 1 shows the results of inserting 50000 records. The table includes the insert execution time to SQL Server. However, this time is very small compared to the time taken to insert the same number of assets into BigchainDB, which took approximately 20 continuous hours (71,846.1 seconds), during which the assets were inserted without interruption. Figures 16 and 17 show that SQL Server has higher CPU consumption compared to the consumption values in BigchainDB. Figures 18 and 19 show that SQL Server has higher memory consumption compared to the consumption values in BigchainDB. It is observed that significant resource consumption increases with the increase in the number of records.

4.1.5 Case 5- Insert 100000

The results of inserting 100000 assets showed high resource consumption compared to previous insertion operations on the BigchainDB network. Table 1 illustrates that the execution time is large, almost double compared to the execution time of 50000 assets. It was about 43 hours during which assets were generated continuously without interruption. Data generation started on August 1 at 7:07 AM and continued without interruption until the 100000 assets were generated on August 3 at 1:49 AM. The code computation time showed the execution time (153905.3 seconds). The SQL Server input process took the longest time for 100000 records among all the previous cases, but the resource consumption is high. The input time taken by BigchainDB is very large compared to that taken by SQL Server. Figures 20 and 21 show that SQL Server has higher CPU consumption compared to the consumption values in BigchainDB. Figures 22 and 23 show that SQL Server has higher memory consumption compared to the consumption values in BigchainDB. It is observed that significant resource consumption increases with the increase in the number of records.

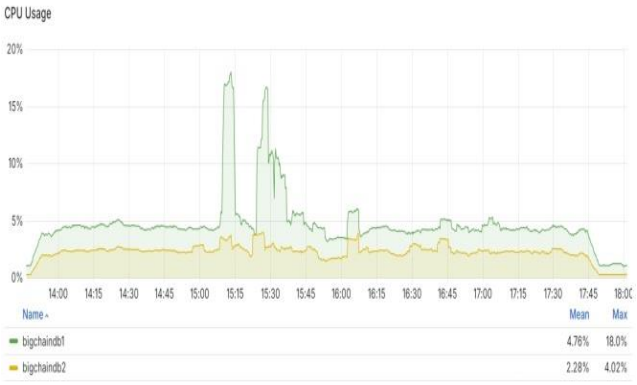


Figure 12. 10K inserts CPU usage BigchainDB node1&2

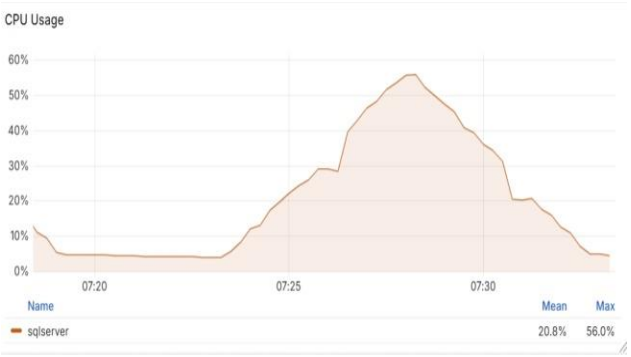


Figure 13. 10K inserts CPU usage SQL server

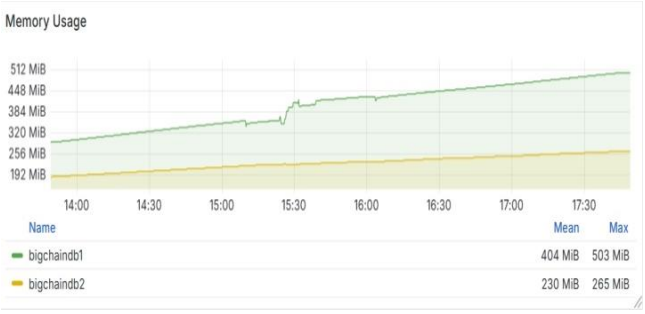


Figure 14 10K inserts memory usage BigchainDB node1&2

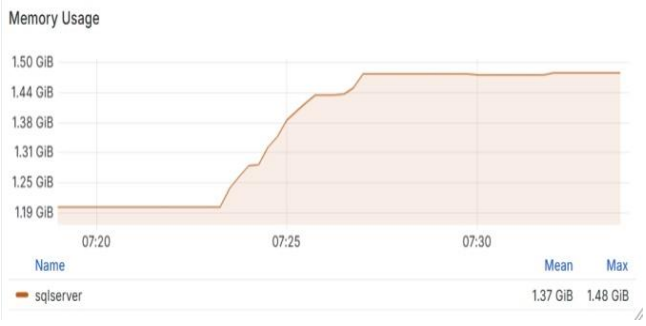


Figure 15 10K inserts memory usage SQL server

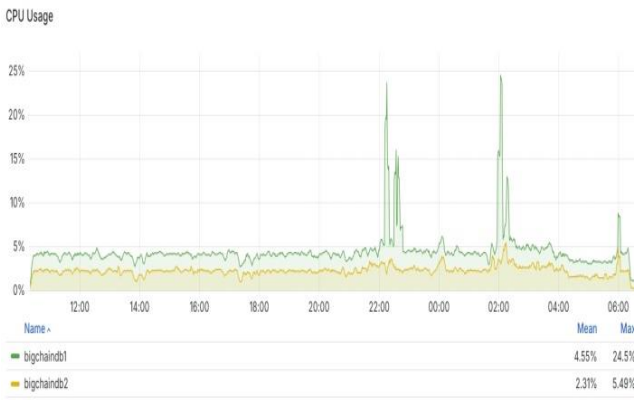


Figure 16 . 50K inserts CPU usage BigchainDB node1&2

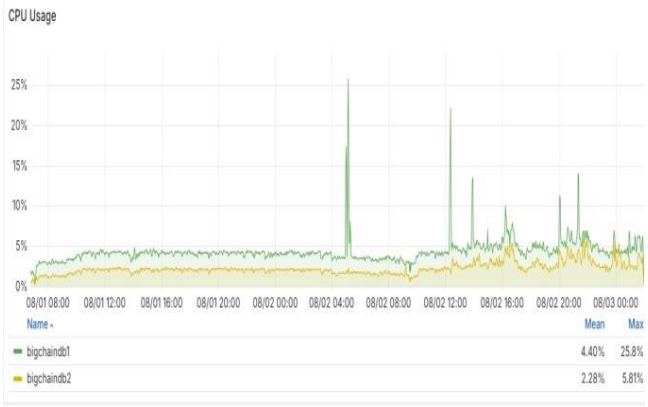


Figure20:100K inserts CPU usage BigchainDB node1&2

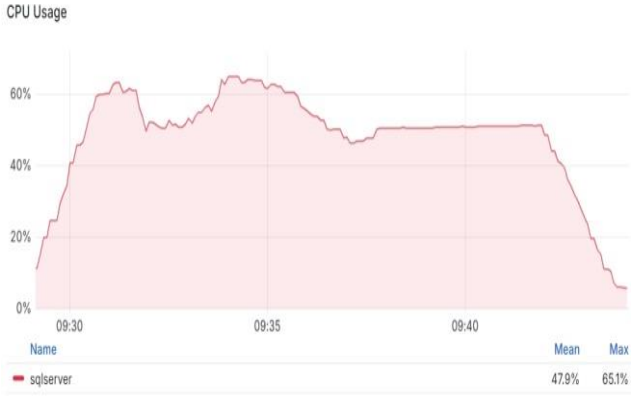


Figure 17. 50K inserts CPU usage SQL server

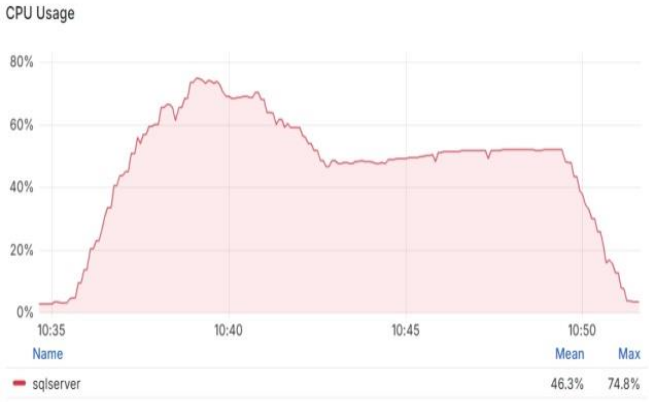


Figure 21. 100K inserts CPU usage SQL server

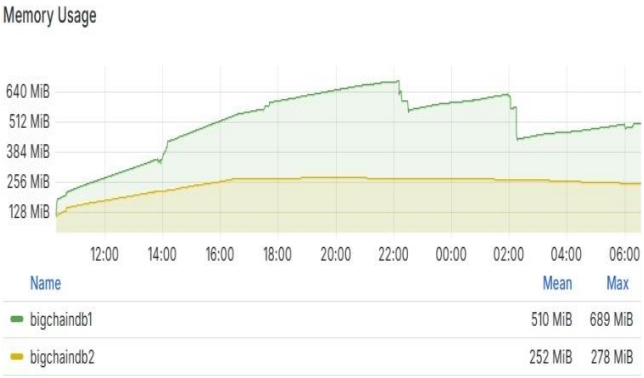


Figure 18. 50K inserts memory usage BigchainDB node1&2

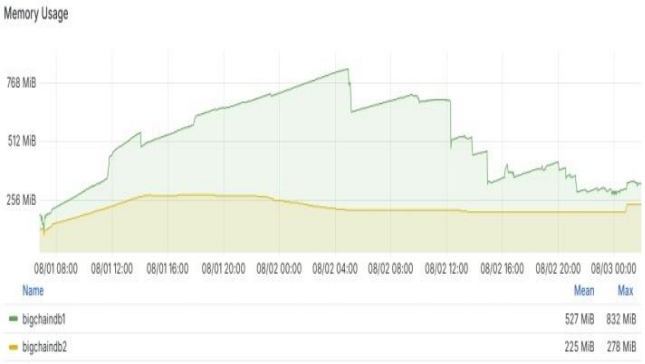


Figure 22. 100K inserts memory usage BigchainDB node1&2

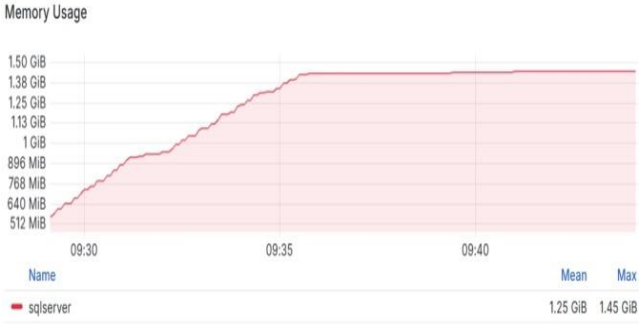


Figure 19. 50K inserts memory usage SQL server

Figure 23. 100K inserts memory usage SQL server

4.2. Query operation Performance

To evaluate the efficiency of both systems in data retrieval, a set of queries was executed under identical conditions. The queries included exact match searches using a single element, such as the public key or asset ID in BigchainDB or the record ID in SQL Server, and patent type searches in both systems. Each query was executed using Python scripts, with both query response time and resource usage measured. Table 2 shows the execution times for both systems with their system resource consumption. SQL Server demonstrated faster performance over BigchainDB, but still had higher CPU and memory consumption. The results show that the execution time of BigchainDB is higher than that of SQL Server for this query operation. Figures 24-33 show the results that highlight the powerful indexing and query execution capabilities of SQL Server compared to BigchainDB that relies on transaction chain traversal, which takes more time to process transactions under consensus mechanism and transaction sequencing requirements.

4.3. Functional Evaluation Metrics:

4.3.1 Immutability

We discuss the ability of both BigchainDB and SQL Server to support record updates and deletes, which are essential operations in any data management environment. SQL Server, as a traditional relational database management system, supports complete updates and deletes automatically via SQL syntax. These operations were successfully implemented and measured in terms of execution time and system resources. The results showed that SQL Server handled these operations efficiently and at the lowest cost.

In contrast, BigchainDB does not allow these operations, which is consistent with Blockchain immutability. Direct deletion or modification of assets is not permitted. Instead, any state change is achieved by issuing a new transfer transaction that references the previous asset. This provides us with traceability, another feature of Blockchains. In contrast, BigchainDB does not allow these operations, which is consistent with Blockchain immutability.

Table 2: Results of the queries operation in BigchainDB and SQL server

Queries By		BigchainDB				SQL Server	
		Node 1		Node 2			
		Mean	Max	Mean	Max	Mean	Max
patent type	CPU usage (%)	2.06%	2.47%	2.32%	2.86%	3.63%	4.58%
	Memory usage (MB)	190MB	190MB	188MB	275MB	833MB	833MB
	Time in second	1.78S		2.17S		0.0303S	
	NO of Assets	43727				43292	
Asset/ record ID	CPU usage (%)	1.12%	1.40%	1.62%	1.93%	3.70%	3.83%
	Memory usage (MB)	223MB	223MB	201MB	201MB	843MB	843MB
	Time in second	0.24S		0.15S		0.010593S	

Figure 24. Query by type CPU usage in BigchainDB node 1&2

Figure 25. query by Type CPU usage in SQL server

Figure 26. query type memory , BigchainDB node1, node2

Figure 27. Query Type memory, SQL server

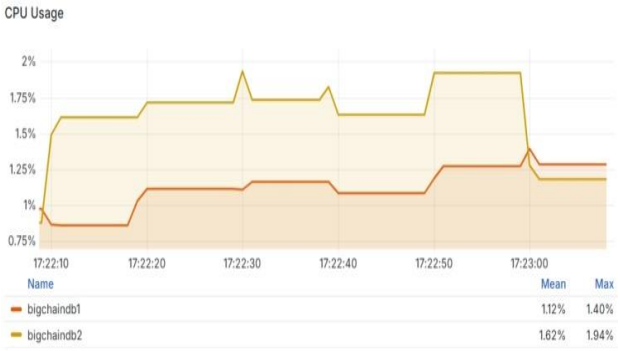


Figure 28. query by asset ID CPU usage in BigchainDB node1&2

Figure 29. Query by record ID CPU usage in SQL server

Figure 30. Query by asset ID memory usage in BigchainDB node1&2

Figure 31. Query by record ID memory usage in SQL Server

```
Enter the original (sender's) public key:
BYKzw45tdaNCMvPZNwAopKXUKPVCiCaKJErYAeVfXTJX
Enter the original (sender's) private key: G7S15w22mxqUcCkRgp4egR11jDT1jWpBVnyrhp3A1eLW
Enter the recipient's public key: F7NwNsZCP1seLrKy5kj8cP2PNoXW7CUxwPYcf3mCsDLx
Enter the transaction ID of the original asset:
bed15616b1073786d309ff7dd3eadf549c2f73f0bd092d98afc8b937385ea671
!Asset transferred successfully
New Transaction ID: 963a0a56d843571de933f362bf9a53464b08fa99000c0876263f0d257824158a
New Owner Public Key: F7NwNsZCP1seLrKy5kj8cP2PNoXW7CUxwPYcf3mCsDLx
 Time taken to transfer asset: 1.6082 seconds
```

Figure 32. An example of a transaction asset using a private key

```
bigchaindb_driver.exceptions.BadRequest: (400,
'{"message":"Invalid transaction (DoubleSpend): input
`bdc63f9c2f6c887b1c19f3849c0158c4a23f1be68e1319
dcf085c5a1aa2c65e7` was already
spent","status":400}\n',{'message': 'Invalid transaction
(DoubleSpend): input
`bdc63f9c2f6c887b1c19f3849c0158c4a23f1be68e1319
dcf085c5a1aa2c65e7` was already spent', 'status': 400},
'http://localhost:9984/api/v1/transactions/')
```

Figure 33. An error message appears when resending the asset using the same private key

```
Transaction Details:
 Transaction D :
7731212bf35317299c8fd8de8de5f7320bbb405e53dbbdc06d40bbca8558d6e7
Operation : TRANSFER
 Asset ID :
bdc63f9c2f6c887b1c19f3849c0158c4a23f1be68e1319dcf085c5a1aa2c65e7
 Previous Owner : 68k2tkcpzhyBnm9Gb68GAEHNpFT327Dh93SivUCzphyK
Current Owner : 3dq7igmCQJseFgjsJp3BEk9Rx9tBa3SiGCQaFFcKWHK8
This transaction is a TRANSFER from previous transaction ID:
↪ bdc63f9c2f6c887b1c19f3849c0158c4a23f1be68e1319dcf085c5a1aa2c65e7
```

Figure 34. An example of a search by Transaction ID shows the transaction details.

Direct deletion or modification of assets is not permitted. Instead, any state change is achieved by issuing a new transfer transaction that references the previous asset. This provides us with traceability, another feature of Blockchains. Table 3 summarizes the extent to which both systems support these operations and the impact of update/delete operations on resources in each. It's important to note that the delete and modify operations in BigchainDB are not supported by the system. The time in the table for both operations (delete and update) was calculated using the same method as for the other operations in this study within Python. However, since the delete and modify operations were not executed, the time shown in Table 3 for these operations is the time returned after the execution attempt failed. The system that initiated the connection to the database should return a message indicating that the operation is not possible for that system. From a functional evaluation perspective, SQL Server offers a broader and more flexible processing model. BigchainDB, supported by immutability and traceability, is more suitable for systems where security and tamper resistance are of high importance. It is important to clarify that deletion and modification operations differ fundamentally between SQL Server and BigchainDB due to their underlying design differences. . In this study, the immutability of BigchainDB was intentionally, Evaluated

As part of a functional comparison, rather than as a direct equivalent of SQL deletion. BigchainDB focuses on permanent data, asset portability, and ownership tracking, where updates are represented by new transactions rather than actual deletion. Therefore, the comparison highlights the architectural and functional differences between traditional relational databases and Blockchain-based databases. The immutability of records in BigchainDB should be interpreted as a system design feature that supports auditability and asset integrity.

4.3.2 Traceability and Ownership Control:

The owner's ability to control the asset is explained. Only the owners can transfer their asset to another owner because they alone possess the private key. The transfer process requires a signature using the private key, as shown in Figure 32. Once this process is complete, the transferor becomes the previous owner of the asset, and the recipient becomes the current owner and new controller.

The asset remains traceable, revealing its previous owners, but they cannot regain control. If someone attempts to transfer the asset back to another owner, they will be unsuccessful because they will receive an error message stating that it has already been transferred and has a new owner. Only the original owner can perform this transfer again, as shown in Figure 33. This provides greater confidence, security, and traceability.

Table 3: Results of the update and delete operation in BigchainDB and SQL server

		SQL Server		BigchainDB				
				Node 1		Node 2		**NOTE**
		Mean	Max	Mean	Max	Mean	Max	
Update	CPU usage	4.39%	4.66%	0.474%	0.538%	0.279%	0.311%	Asset can't be updated
	Memory usage	843MB	843MB	135MB	135MB	119MB	119MB	
	Time in (S)	0.17313S		0.0090S				
Delete	CPU usage	4.07%	4.43%	0.4.27%	0.491%	0.244%	0.269%	Asset can't be Deleted
	Memory usage	831MB	831MB	135MB	135MB	119MB	119MB	
	Time in (S)	0.0983		0.0114S				

Table 4: Results of Transaction one asset operation and query by this Transaction ID

		BigchainDB			
		Node 1		Node 2	
		Mean	Max	Mean	Max
Transaction Asset	CPU usage (%)	1.24%	1.46%	0.853%	1.00%
	Memory usage (MB)	206MB	206MB	172MB	172MB
	Time in second	1.6082s			
Query by Transaction ID	CPU usage (%)	1.73%		1.94%	
	Memory usage (MB)	226MB		227MB	
	Time in second	0.30		0.36	

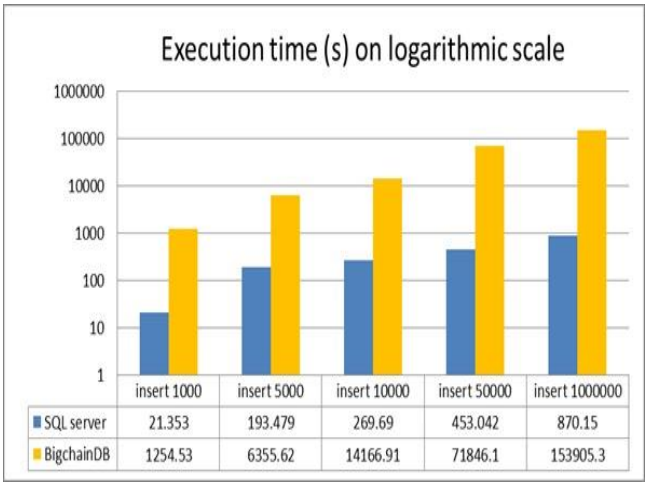
person. This process is done by the owner only by using owner's private key. The key is generated for the first time when the asset is created, and the owner keeps it securely.

The transaction ID search was used to illustrate this feature in BigchainDB. The search using the transaction ID shows whether the asset has a previous or current owner. Figure 34 shows that the asset can be traced even if ownership hasn't changed, which is reflected in the data. Table 4 shows the results of time and Grafana's monitoring of memory and CPU consumption during the operations.

4.4 Discussions:

The results show that SQL Server outperforms BigchainDB across all dataset sizes for execution time, as shown in Figure 35 for the insert operation, with full support for all basic operations. However, as the dataset size increases, the performance gap becomes more pronounced. At 100,000 records, SQL Server completes insertion significantly faster than BigchainDB. The results show high consumption of system resources, including memory and CPU, by SQL Server compared to BigchainDB, as shown in Figures 36 and 37. BigchainDB exhibited low CPU and memory consumption, but exhibited high execution times (Figures 35, 36, 37). Although BigchainDB shows slower insertion speed, this overhead provides stronger guarantees of immutability and trust, which may be desirable in applications requiring tamper-resistant storage. The above behavior can be explained by architectural differences. SQL Server performs direct writes to indexed tables using optimized storage engines, whereas BigchainDB requires multiple additional steps, including transaction creation, cryptographic signing, validation, and consensus agreement among nodes before committing data to the ledger. These processes introduce unavoidable latency and reduce throughput. SQL Server exhibits higher CPU utilization across most workloads. This is expected because relational systems optimize performance using indexing, caching, and query processing algorithms that actively consume computational resources.

SQL Server consumes more memory than BigchainDB, due to buffer pools, caching mechanisms, and indexing structures designed to accelerate query execution. These optimizations improve performance but increase resource requirements.



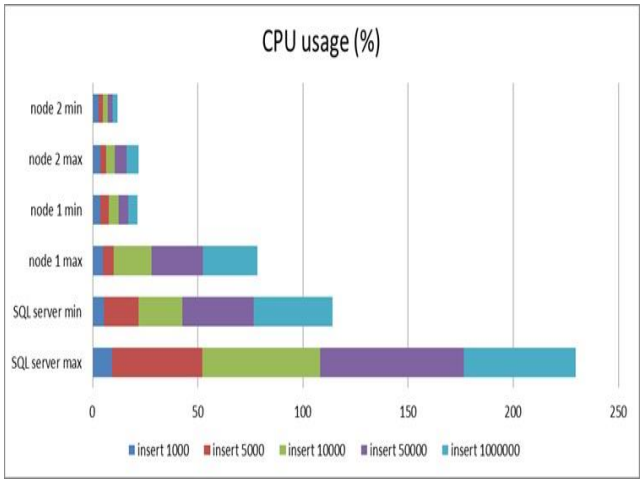


Figure 35. Execution time(s) on logarithmic scale for insert operation

Figure 36. CPU usage (%) for insert operation

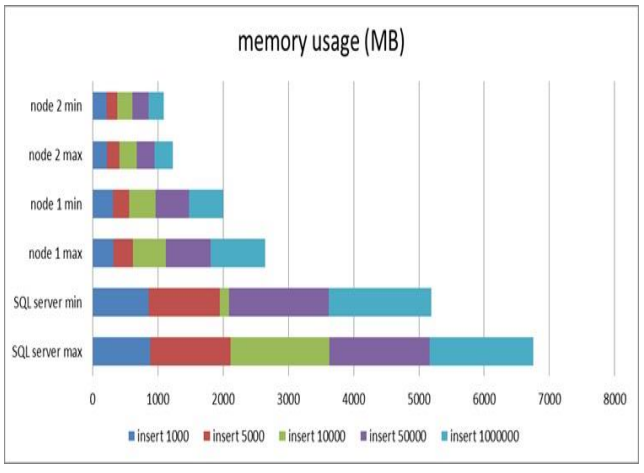


Figure 37. Memory usage (MB) for insert operation

BigchainDB maintains relatively stable and lower memory usage, reflecting its simpler storage operations and document-based backend. For resource-constrained environments, BigchainDB may offer advantages in terms of a lower memory footprint, though with reduced

performance. The experimental results demonstrate a clear trade-off between centralized and decentralized database architectures. SQL Server consistently delivers superior insertion and query performance due to its centralized design, mature indexing mechanisms, and optimized execution engine. However, this performance comes at the cost of higher CPU and memory consumption and reliance on a trusted central authority. In contrast, BigchainDB provides enhanced security properties, including immutability, traceability, and decentralized control. These features make it suitable for applications such as digital asset management, supply chain tracking, and record auditing, where data integrity is more critical than raw speed. However, consensus mechanisms and cryptographic operations introduce latency that limits its suitability for high-frequency transactional systems

Conclusion

In this research paper, we demonstrated the potential use of Blockchain technologies in data management, focusing on BigchainDB, a Blockchain-based data management system. We conducted a comparative analysis of BigchainDB against a commonly used traditional relational database, SQL Server. Implementing identical operations in the same environment, including insert and query. We evaluated their performance in terms of execution time and resource consumption. We also evaluated functional evaluations of information immutability, traceability, and Ownership control.

The results showed that SQL Server outperformed BigchainDB in most operations in terms of execution time. It exhibited lower latency but higher resource consumption, particularly during high-volume inserts and complex queries. SQL Server support for all operations also provided greater flexibility and ease of use. In contrast, BigchainDB, due to its Blockchain oriented architecture and immutable model, demonstrated lower performance in execution speed but also lower resource consumption than SQL Server. It also demonstrated greater tamper resistance, data integrity, and traceability, which may have affected performance. The results of experiments with SQL Server demonstrated better performance and speed according to certain benchmarks; this depends on the context and specific requirements of the application being developed. It may be more suitable for general data management tasks where speed, efficiency, and operational flexibility are prioritized. However, BigchainDB also demonstrated lower resource consumption for most operations, as well as greater strength in ensuring traceability and immutability of data, and the ability of the owner to control assets. This undoubtedly makes many users find BigchainDB attractive and suitable for applications that require these features and bear the costs. Although the methodology used in this study was carefully designed to provide a comparison between SQL Server and BigchainDB, there are some limitations to this framework. The experiments were conducted in a resource-limited environment and with hardware configurations that may not fully reflect large-scale production environments. Furthermore, the data volume processed was still below the scales we hoped to use and systems might encounter in real-world applications. These limitations do not diminish the value of the results, but they clearly define the framework within which they should be interpreted.

Future Work:

While this study focused on a comparison of SQL Server and BigchainDB in terms of performance and functional metrics, this study has certain limitations, and there are still broad areas for future work. The experiments were conducted on a single host with limited hardware resources and a maximum dataset size of 100,000 records. Additionally, only one relational database and one Blockchain-based database were evaluated. Future work will extend this research by testing larger-scale distributed deployments, incorporating additional database platforms, increasing dataset sizes, increasing the number of nodes used in Blockchain-based systems, and using an actual distributed environment to reflect the generalizability of results. Investigating hybrid architectures that combine relational and Blockchain storage may also offer promising directions for achieving both high performance and strong trust guarantees. Overall, this work contributes an empirical foundation for understanding the practical trade-offs between centralized and decentralized database systems and provides guidance for researchers and practitioners when selecting appropriate technologies for secure and scalable data management.

Disclosure

The authors should clear the conflicts of interest in their work.

References

1. S. Wilson et al., 'Data Management Challenges in Blockchain-Based Applications', IEEE Internet Comput., vol. 28, no. 1, pp. 70–80, Jan. 2024,
2. Q. Wei, B. Li, W. Chang, Z. Jia, Z. Shen, and Z. Shao, 'A Survey of Blockchain Data Management Systems', Nov. 25, 2021, arXiv: arXiv:2111.13683. Accessed: Sep. 22, 2024. [Online]. Available: http://arxiv.org/abs/2111.13683
3. P. Chitti, J. Murkin, and R. Chitchyan, 'Data Management: Relational vs Blockchain Databases', in Advanced Information Systems Engineering Workshops, vol. 349, H. A. Proper and J. Stirna, Eds., in Lecture Notes in Business Information Processing, vol. 349. , Cham: Springer International Publishing, 2019, pp. 189–200.
4. Monrat, O. Schelen, and K. Andersson, 'A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities', IEEE Access, vol. 7, pp. 117134–117151, 2019,
5. BigchainDB GmbH, Berlin, Germany ,'BigchainDB 2.The Blockchain Database', https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf, p. 14, 2018.
6. MongoDB. https://www.mongodb.com/
7. T. McConaghy et al., 'BigchainDB: A Scalable Blockchain Database', ascribe GmbH, Berlin, Germany, Jun. 2016.
8. Rustemi, V. Atanasovski, and A. Risteski, 'Framework for Using BigchainDB in Software Application', 2022.
9. T. I. Nurmamatovich, 'The SQL server language and its structure', American Journal of Open University Education, vol. 1, no. 1, 2024.
10. X. Chen, Y. Liu, and J. Ge, 'A Data Management Method Based on Blockchain Technology', in 2020 3rd International Conference on Smart BlockChain (SmartBlock), Zhengzhou, China: IEEE, Oct. 2020, pp. 203–208
11. J. Chen, Z. Lv, and H. Song, 'Design of personnel big data management system based on blockchain', Future Generation Computer Systems, vol. 101, pp. 1122–1129, Dec. 2019
12. S. Lupaiescu, P. Cioata, C. E. Turcu, O. Gherman, C. O. Turcu, and G. Paslaru, 'Centralized vs. Decentralized: Performance Comparison between BigchainDB and Amazon QLDB', Applied Sciences, vol. 13, no. 1, p. 499, Dec. 2022,
13. A. Alotaibi, S. Alissa, and S. Mohammed, 'A comparative study of blockchain data management systems: BIGCHAINDB vs. FALCONDB', International Journal of Computer Science and Network Security, vol. 23, no. 5, pp. 127–133, May 2023,.
14. Y. Wang, C.-H. Hsieh, and C. Li, 'Research and Analysis on the Distributed Database of Blockchain and Non-Blockchain', in 2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), Chengdu, China: IEEE, Apr. 2020, pp. 307–313.
15. A. Malik, A. Burney, and F. Ahmed, "A Comparative Study of Unstructured Data with SQL and NO-SQL Database Management Systems," *J. Comput. Commun.*, vol. 08, no. 04, pp. 59–71, 2020 .
16. A. Rudniy, 'Data Warehouse Design for Big Data in Academia', Computers, Materials & Continua, vol. 71, no. 1, pp. 979–992, 2022,
17. C. Ming Wu, Y. Fu Huang, and J. Lee, 'Comparisons Between MongoDB and MS-SQL Databases on the TWC Website', AJSEA, vol. 4, no. 2, p. 35, 2015,
18. Docker . https://www.docker.com/
19. N. A. Sultan and R. Putros Qasha, "CONTAINER-BASED VIRTUALIZATION FOR BLOCKCHAIN TECHNOLOGY: A SURVEY," Jordanian J. Comput. Inf. Technol. JJCIT, vol. 9, no. 3, Sept. 2023.
20. Faker Python package /project/Faker/
21. BigchainDB . https://www.bigchaindb.com

University of Aden Journal of Natural and Applied Sciences., vol.29(2) (2025) 61-76

مجلة جامعة عدن للعلوم الطبيعية والتطبيقية

Journal homepage: https://uajnas.adenuniv.com

بحث علمي

مقارنة تجريبية ووظيفية بين BigchainDB و SQL Server لإدارة البيانات

ماريا عثمان صالح مقشع , خالد احمد عبود عمر
قسم علوم وهندسة الحاسوب - كلية الهندسة، جامعة عدن

https://doi.org/10.47372/uajnas.2025.n2.a07

الملخص

أدى النمو المتسارع للبيانات الرقمية إلى زيادة الطلب على أنظمة إدارة البيانات التي لا توفر فقط الأداء العالي وقابلية التوسع، بل توفر أيضًا ضمانات قوية للأمان والنزاهة والموثوقية. تقدم هذه الورقة مقارنة تجريبية ووظيفية بين SQL Server، وهي قاعدة بيانات علائقية تقليدية، وBigchainDB، وهي قاعدة بيانات لامركزية قائمة على تقنية Blockchain التي تدمج ميزات السجلات الموزعة مع إمكانيات قواعد البيانات. تم تنفيذ كلا النظامين في بيئة محاكاة متطابقة باستخدام Docker لضمان تقييم عادل وقابل للتكرار. تم إنشاء مجموعة بيانات موحدة تحتوي على ما يصل إلى 100,000 سجل، واستُخدمت لتقييم أداء الإدخال، وزمن استجابة الاستعلام، وقابلية التوسع، واستهلاك موارد النظام (وحدة المعالجة المركزية والذاكرة). تمت مراقبة سلوك النظام باستمرار باستخدام Prometheus وGrafana. بالإضافة إلى مقاييس الأداء، تم تقييم المقاييس الوظيفية، بما في ذلك عدم قابلية التغيير(الثبات)، وإمكانية التتبع، والتحكم في الملكية.

تُظهر النتائج التجريبية أن SQL Server يحقق زمن استجابة أقل بكثير واستجابة أسرع للاستعلامات، ولكن على حساب زيادة استهلاك وحدة المعالجة المركزية والذاكرة. في المقابل، يُظهر BigchainDB استهلاكًا أقل للموارد، ويُوفر أمانًا قويًا وضمانات مقاومة للتلاعب، مع زيادة في زمن الاستجابة نتيجة لآليات الإجماع والتحقق من صحة المعاملات. تُسلط هذه النتائج الضوء على المفاضلة بين حلول إدارة البيانات المركزية واللامركزية، وتُقدم إرشادات عملية لاختيار التقنية المناسبة بناءً على متطلبات التطبيق من حيث الأداء والموثوقية والأمان.

مفاتيح البحث

التسليم : 05/01/ 2026

القبول: 19/02/ 2026

كلمات مفتاحية :

Blockchain إدارة البيانات، SQL Server، BigchainDB، تقييم الأداء، قواعد البيانات اللامركزية، التقييمات الوظيفية.