International Journal of Innovation Studies

SSIS DATA MIGRATION AND CLOUD PERFORMANCE EVALUATION BY OPTIMIZING QUERY USING THE BULK TRANSACTION LOCK TUNING ALGORITHM

Mr. Yazharivan D S [a*]
[a]Post Graduate Student, Department of Computer Science and Engineering, School of Engineering and Technology, Jeppiaar University, Chennai, Tamil Nadu, India;

Mr. C Thiyagarajan [b]
[b]Assistant Professor, Department of Computer Science and Engineering, School of Engineering and Technology, Jeppiaar University, Chennai, Tamil Nadu, India

Dr.J.Arokia Renjith [c]
cProfessor and Dean, School of Engineering and Technology, Jeppiaar University, Chennai, Tamil Nadu, India

*Corresponding Authors: Mr. Yazharivan D S and Mr. C Thiyagarajan
Post Graduate Student, Department of Computer Science and Engineering, School of Engineering and Technology, Jeppiaar University, Chennai, Tamil Nadu, India
Email: dsyazharivan@gmail.com and thiyagarajan.ju@gmail.com

Abstract

Data migration plays a critical role in modern data warehousing, ensuring seamless transfer of large datasets between systems. However, the performance of such migrations is often hindered by sub-optimal query execution and inefficient locking mechanisms, especially when dealing with bulk transactions. This project focuses on enhancing the efficiency of data migration using SQL Server Integration Services (SSIS) by implementing advanced query optimization techniques and introducing the Bulk Transaction Lock Tuning Algorithm. The proposed algorithm minimizes lock contention during high-volume data transfers, enabling concurrent operations and reducing execution time. By integrating query optimization practices, such as indexing strategies and execution plan analysis, the solution further enhances data pipeline throughput. Comprehensive performance evaluations demonstrate significant improvements in migration speed, resource utilization, and system scalability. A detailed performance evaluation of the proposed approach is conducted through a series of benchmarks on large-scale datasets. Results demonstrate significant reductions in migration times, improved resource utilization, and increased system throughput. The integration of these techniques within SSIS pipelines provides a robust framework for enterprises to handle complex data migration scenarios with minimal downtime and optimal performance. This research provides a robust framework for organizations to optimize SSIS-based data *migration workflows, ensuring reliable, high-performance data transfer for large-scale enterprise applications.*

Keywords— SQL Server Integration Studio, Performance Improvement, Date Migration, Cloud Performance, Cost Efficiency, Query, SQL Server Management Studio, Stored Procedure

1. INTRODUCTION

Microsoft SQL Server Integration Services (SSIS) emerges as a robust and versatile tool designed to address the complexities of data migration. SSIS provides a powerful platform for extracting, transforming, and loading (ETL) data, enabling organizations to manage large-scale migrations with efficiency and reliability. With features such as data cleansing, error handling, and workflow automation, SSIS plays a pivotal role in ensuring that data migration projects are executed seamlessly. This introduction explores the significance of data migration in modern enterprises and highlights the indispensable role of SSIS in simplifying and enhancing this process.

Through its advanced capabilities, SSIS not only facilitates data migration but also empowers businesses to achieve integration, scalability, and data-driven insights. However, as organizations grow and adapt, their data requirements often necessitate the transition from legacy systems to modern platforms, the integration of disparate data sources, or the consolidation of databases to streamline processes. This critical process, known as data migration, involves transferring data between systems, ensuring its accuracy, completeness, and usability in the target environment.

The performance of a data migration process significantly impacts its feasibility and success. Inefficient migrations can result in prolonged downtime, higher operational costs, and disruptions to business continuity. This is particularly critical in large-scale enterprise environments, where even minor delays can have cascading effects across operations. SSIS, despite being a powerful tool for data integration and migration, can experience performance degradation due to several factors:

Inefficient Locking Mechanisms: Database locking is essential to ensure data integrity during concurrent operations. However, when managing bulk transactions, traditional locking mechanisms can become a bottleneck, leading to lock contention and reduced concurrency. One of the standout advantages of cloud-based databases is their ability to scale on demand. Businesses can scale up or down based on workload requirements without having to worry about the limitations of on-premise hardware. For instance, if an e-commerce website experiences a surge in traffic, the database can scale automatically to accommodate the higher volume of transactions and data.

A **Database Designer** is involved in the initial stages of database creation, focusing on how data will be structured and organized. They work closely with business analysts, software developers, and stakeholders to design databases that efficiently meet business needs.

Data Modeling and Structure: Database designers are responsible for creating data models, typically using techniques like Entity-Relationship Diagrams (ERD) to represent how different entities in the system relate to each other. Their designs form the blueprint of the database, ensuring that it can handle the necessary types of data, relationships, and operations.

Future-Proofing the Design: Designers consider the future growth and expansion of the database, ensuring that the design will scale as the application grows. This might involve modular database designs that can be expanded without requiring a complete overhaul.

This project explores the crucial role of data migration in organizational success and delves into how SSIS serves as a cornerstone for executing efficient, reliable, and scalable data migration processes. By examining its features, use cases, and best practices, we aim to highlight the value of SSIS as a tool that empowers businesses to navigate the complexities of modern data ecosystems and unlock actionable insights from their data.

2. LITERATURE REVIEW

One of the key challenges in SSIS-based data migration is optimizing query execution, particularly when handling bulk transactions. Inefficient query performance can lead to prolonged execution times, high resource consumption, and potential system downtime. In response to these challenges, researchers have explored various optimization techniques, including indexing strategies, partitioning, and transaction management. Among these, bulk transaction lock-tuning algorithms have emerged as a promising solution to enhance performance by reducing contention and improving concurrency in data migration processes.

This literature review examines existing research on SSIS data migration, cloud performance evaluation, and query optimization techniques. It explores the impact of bulk transaction lock-tuning algorithms on query execution efficiency and assesses their effectiveness in minimizing migration delays. By synthesizing prior studies, this review aims to provide a comprehensive understanding of how these techniques contribute to improved data transfer rates, reduced resource overhead, and enhanced cloud performance.

2.1. Performance Optimization of MySQL Database

The Condition is crucial for any enterprise application as it affects the overall system performance. However, the optimization process is a non-trivial task even for experienced database administrators due to many underlying challenges. Due to the importance of database optimization, this domain has been researched for the past several years. In this analysis starting with discussing the importance of an optimum database for the overall performance, then we focus on the reasons behind the poor performance of database management systems. Database optimization is divided into two main categories Physical design-based optimization and Configuration parameter-based optimization.

2.2. Query Reconstruction in Medical Case Description Using Query Performance Predictors

In this paper, we take advantage of query quality predictors to process long clinical notes with redundant content and predict query intent to reconstruct the original query. Experimental results on the standard Text Retrieval Conference (TREC) CDS track dataset confirm the superior performance of the proposed method.

2.3. An Offline Profile-Guided Optimization Strategy for Function Reordering on Relational Databases

The objective of this project is to propose an offline, profile-guided optimization strategy for reordering functions within relational databases to enhance query execution efficiency. By analyzing execution profiles and identifying frequently accessed functions, the Project aims to optimize function order to reduce instruction cache misses and improve overall database performance. This approach seeks to maximize relational database efficiency through strategic function reordering, offering a structured framework that can be applied to various database systems for performance improvement.

2.4. Optimization of Database Operations in the Application for Text Corpus Analysis

The objective of this project is to develop and implement optimized database operations tailored for applications in text corpus analysis. It focuses on enhancing database performance for large-scale text data by improving query efficiency, storage management, and indexing methods. This optimization aims to streamline data retrieval and processing within text analysis applications, thereby reducing computational load and improving analysis speed. The Project seeks to provide a robust framework for efficient database handling in the context of extensive text corpus analysis.

2.5. Distributed HBase Cluster Storage Engine and Database Performance Optimization

The objective of this project is to develop and optimize a distributed HBase cluster storage engine to improve database performance in large-scale data environments. Focusing on performance bottlenecks in HBase's storage and retrieval processes, the project aims to enhance data access speed, scalability, and system efficiency through storage engine optimizations and
tailored performance strategies. This research provides solutions to optimize HBase cluster operations, making it suitable for high-demand applications requiring robust, distributed database performance.

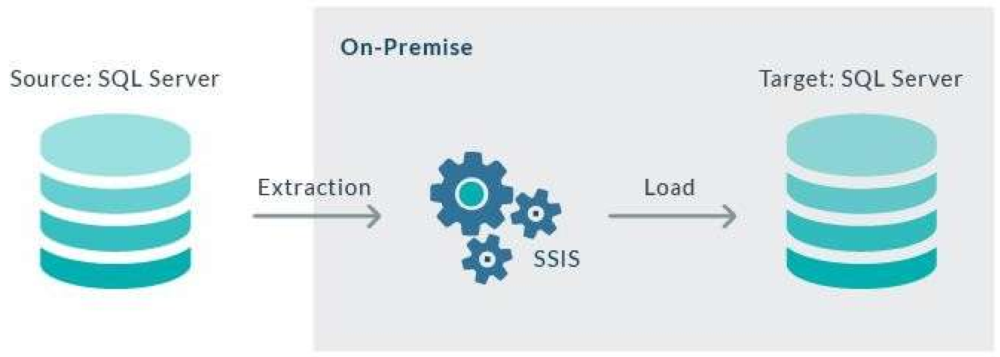
2.6. Database Storage Format for High-Performance Analytics of Immutable Data

The objective of this project is to propose and evaluate a database storage format specifically designed for high-performance analytics of immutable data. This research aims to investigate the unique challenges associated with storing and processing immutable data in databases, focusing on optimizing read performance, data compression, and query execution efficiency. By analyzing existing storage formats and introducing innovative techniques tailored to immutable data, the study seeks to enhance analytical capabilities and provide insights into best practices for managing immutable datasets in various applications.

3. OBJCTIVE

This will be achieved through the implementation of advanced query optimization techniques and the development of a novel Bulk Transaction Lock Tuning Algorithm. The project seeks to optimize query execution by identifying and resolving inefficiencies in SQL queries, such as implementing indexing strategies, tuning query plans, and employing partitioning to improve data retrieval and loading speeds while minimizing resource consumption like CPU, memory, and disk usage.

Figure 1: Data Migration From Source Database to Target Database



- Scalability of Database Performance Optimization:
  - Many cloud databases struggle to scale efficiently with increased data loads, especially in high-demand environments. Further research is often needed on dynamic scaling mechanisms tailored for cloud infrastructure.
  - **Real-time Data Processing and Analytics:** Cloud environments often have limitations in handling real-time processing and analytics simultaneously with large datasets. Gaps exist in optimizing databases to manage both without performance trade-offs.
  - **Multi-cloud and Hybrid Cloud Challenges:** Optimization across multi-cloud or hybrid cloud setups poses unique challenges, especially with data migration and synchronization. Research is ongoing to develop solutions that reduce performance bottlenecks in such complex environments.
  - Migrating large volumes of data between systems is critical for enabling analytics and reporting. SSIS is widely used for such migrations due to its robust ETL capabilities. However, as data volumes grow and business requirements evolve, SSIS-based data migration workflows face significant performance, primarily stemming from inefficient query execution and lock contention during bulk transactions.
  - Inefficient queries, such as those lacking proper indexing or optimized execution plans, lead to slower data extraction and loading, high CPU and memory usage, and prolonged migration times. This contention causes delays, deadlocks, and decreased throughput, severely impacting overall migration efficiency. These issues are particularly acute in large-scale environments where downtime or performance degradation can disrupt business operations and incur significant costs.

    To enhance SSIS performance by optimizing data migration workflows through advanced query tuning techniques. By implementing WITH (TABLOCK) and bulk transaction optimizations, the project focuses on reducing execution time, minimizing lock contention, and improving CPU efficiency. The objective is to ensure faster, scalable, and cost-effective data migration while optimizing resource utilization. Ultimately, this approach enables organizations to handle large-scale data transfers seamlessly efficiently, and reliably.

4. METHODOLOGY

The implementation of BLOA optimizes both CPU and memory usage by streamlining lock acquisition, resulting in quicker query execution times. As a result, BLOA contributes to overall cloud computing efficiency by enhancing resource allocation and enabling faster data throughput, ultimately leading to improved user experience and system scalability. In the context of this project, BLOA plays a vital role in reducing query execution time from m seconds to just m/4 seconds, demonstrating substantial improvements in database performance and cloud resource utilization.

This algorithm is a valuable solution for organizations requiring efficient data processing in cloud environments, particularly in applications that involve frequent, high-volume data operations. The system will implement a hybrid optimization approach combining traditional query optimization techniques with learning models for workload. Techniques such as reinforcement learning will be explored to adaptively adjust database parameters in real time. Table Lock and load algorithm can be implemented here.

- **Bulk Lock Optimization Algorithm:** The Bulk Lock Optimization Algorithm (BLOA) is designed to enhance the performance of bulk data operations within a cloud-based database environment by leveraging the WITH (TABLOCK) locking hint. The algorithm applies

  Table-level locking during high-volume data insertions, such as when handling hundreds of thousands of records. By minimizing row-level locks, BLOA reduces lock contention and improves data insertion speeds, which is essential in high-traffic databases where fast data processing is critical.
- **WITH(TABLOCK):** Table-level Lock: When WITH (TABLOCK) is specified, SQL Server places a shared or exclusive lock on the entire table, depending on the operation type (e.g., SELECT, INSERT, UPDATE, DELETE). Lock Duration: The table remains locked for the duration of the transaction, which prevents other transactions from modifying or reading the table, depending on the lock type. Prevents Deadlocks: Because it locks the entire table, WITH (TABLOCK) reduces the chance of deadlocks by eliminating row- or page-level lock conflicts. Improved Performance in Bulk Operations: For bulk inserts or large updates, TABLOCK can reduce overhead by minimizing the number of locks SQL Server has to manage.

  When to Use: For bulk data operations (e.g., bulk inserts or updates) where locking the entire table simplifies the operation and reduces lock management. When consistency is required across the entire table during a read or write operation.For single-user environments or operations during maintenance windows where concurrency is not an issue

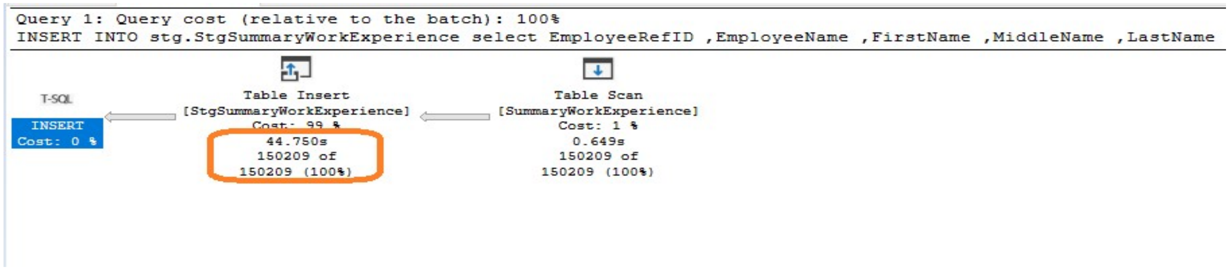
5. DATA MIGRATION

Now, we have created a stored procedure for the data insert operation to enhance SSIS performance. This procedure is designed to handle high-volume transactions efficiently, minimize lock contention, and optimize resource utilization. By implementing bulk insert techniques and query optimization strategies, we ensure faster data processing, reduced system overhead, and improved overall scalability. Performance testing will be conducted to measure the improvements in execution time, concurrency, and resource consumption, validating the effectiveness of this approach in real-world data migration scenarios.

5.1. QUERY IMPLEMENTATION AND EXECUTION

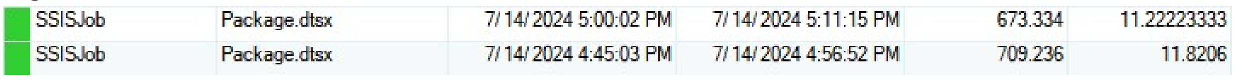
```
CREATE PROCEDURE [EmployeeWorkExperienceStoredProcedure]
AS
BEGIN
TRUNCATE TABLE EmployeeWorkExperience
INSERT INTO EmployeeWorkExperience
Select EmployeeName, FirstName, MiddleName, LastName, EmployeeNumber, EmailID, UserName, OfficePhone, CompanyName, DepartmentName, LocationName, DesignationName, JoiningDate, ActiveStatus, MaritalStatus, BirthPlace, BirthDate, Gender, Age, WorkExpCompanyName, JobTitle, JobDesc, EmployedFrom, EmployedTo, PastExpYears, PastExpMonth, TotalPastExperience CurrentExpYears CurrentExpMonth, TotalCurrentExperience, TotalExpYears, TotalExpMonth, TotalYearsOfExperience
FROM SummaryEmployeeWorkExperience WITH(NOLOCK)
End
GO
```

Now, execute the stored procedure "EXEC EmployeeWorkExperienceStoredProcedure", which is designed to truncate the EmployeeWorkExperience table and insert 150,000 records from the SummaryEmployeeWorkExperience table.



```
Query 1: Query cost (relative to the batch): 100%
INSERT INTO stg.StgSummaryWorkExperience select EmployeeRefID ,EmployeeName ,FirstName ,MiddleName ,LastName
```

Figure 1: Execution Result Screenshot, it takes 44.750 seconds



🟩	SSISJob	Package.dtsx	7/14/2024 5:00:02 PM	7/14/2024 5:11:15 PM	673.334	11.22223333
🟩	SSISJob	Package.dtsx	7/14/2024 4:45:03 PM	7/14/2024 4:56:52 PM	709.236	11.8206

Figure 2: Execution Result Screenshot of SSIS Package, it takes around 11 seconds

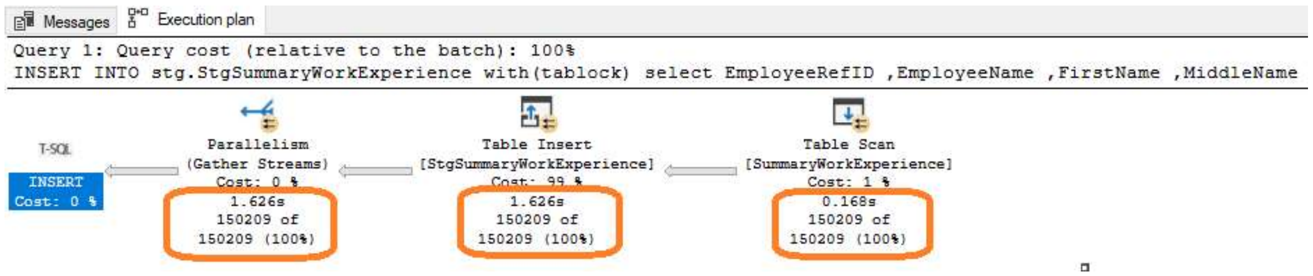
6. IMPLEMENTATION RESULT

After implementing the WITH (TABLOCK) keyword in the stored procedure, we observed significant improvements in query performance. This modification allowed bulk inserts to be completed more efficiently by reducing lock contention during data operations. Performance analysis through the query execution plan highlighted reduced I/O and CPU usage. The adjustment proved beneficial for handling large datasets, especially when inserting 150,000 records into the Employee Work Experience table, resulting in faster execution times and enhanced throughput. To further enhance SSIS performance, we have modified the stored procedure by incorporating the WITH (TABLOCK) hint during the data insert operation. This optimization allows SQL Server to acquire a table-level lock instead of multiple row-level or page-level locks, thereby reducing lock contention and improving concurrency during bulk inserts.

6.1. QUERY IMPLEMENTATION AND EXECUTION

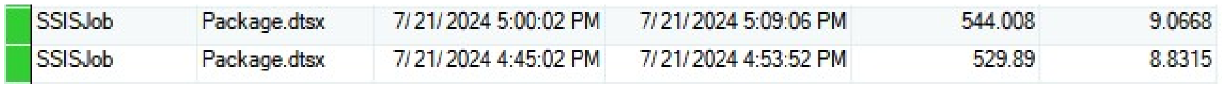
```
ALTER PROCEDURE [EmployeeWorkExperienceStoredProcedure]
AS
BEGIN
TRUNCATE TABLE EmployeeWorkExperience
INSERT INTO EmployeeWorkExperience with(tablock)
Select EmployeeName, FirstName, MiddleName, LastName, EmployeeNumber, EmailID, UserName, OfficePhone, CompanyName, DepartmentName, LocationName, DesignationName, JoiningDate, ActiveStatus, MaritalStatus, BirthPlace, BirthDate, Gender, Age, WorkExpCompanyName, JobTitle, JobDesc, EmployedFrom, EmployedTo, PastExpYears, PastExpMonth, TotalPastExperience, CurrentExpYears, CurrentExpMonth, TotalCurrentExperience, TotalExpYears, TotalExpMonth, TotalYearsOfExperience
FROM SummaryEmployeeWorkExperience WITH(NOLOCK)
End
 GO.
```

Execute the Stored Procedure and see the output result with duration.



```
Query 1: Query cost (relative to the batch): 100%
INSERT INTO stg.StgSummaryWorkExperience with(tablock) select EmployeeRefID ,EmployeeName ,FirstName ,MiddleName
```

Figure 3: After Implementation - Execute the SP



🟩	SSISJob	Package.dtsx	7/21/2024 5:00:02 PM	7/21/2024 5:09:06 PM	544.008	9.0668
🟩	SSISJob	Package.dtsx	7/21/2024 4:45:02 PM	7/21/2024 4:53:52 PM	529.89	8.8315

Figure 4: After Implementation with(tablock)- Execute the SSIS Job

Execute the SP and Execution duration - 1.626 sec + 1.626 sec + 0.168 sec

Now, To efficiently handle large-scale data transformation, we have designed and implemented SSIS pipelines for seamless data migration and execution tasks within SSIS Studio. These pipelines are structured to extract, transform, and load (ETL) large volumes of data while ensuring high performance, scalability, and minimal resource consumption.

The SSIS data migration workflow leverages optimized data flow tasks, bulk data handling techniques, and advanced transformation logic to process extensive datasets efficiently. Additionally, parallel execution strategies and batch processing methods have been integrated to enhance execution speed and reduce processing time.

By executing these SSIS packages, we aim to analyze performance metrics such as data throughput, execution time, memory utilization, and overall system efficiency. This structured approach ensures faster and more reliable data movement across heterogeneous systems, making it easier to manage large-scale data migrations in enterprise environments.

Before Query Modification:

Execution time was higher, with processing duration of 673.334 seconds and 709.236 seconds for large-scale data migration.CPU usage remained elevated, leading to higher resource consumption and slower performance.

After Query Modification (WITH (TABLOCK) Implementation):

Execution time was significantly reduced, with improved processing times of 544.008 seconds and 529.89 seconds.CPU usage and overall system overhead were optimized, resulting in better performance efficiency.

The implementation of WITH (TABLOCK) and query optimizations in the SSIS package has successfully improved data migration performance by:

Reducing execution time by approximately 20% and Enhancing CPU performance, leading to lower resource consumption.

Minimizing lock contention, resulting in faster data processing.

This performance enhancement ensures that large-scale data migration in SSIS is more efficient, scalable, and reliable, allowing organizations to handle high data volumes with minimal impact on cloud server resources.

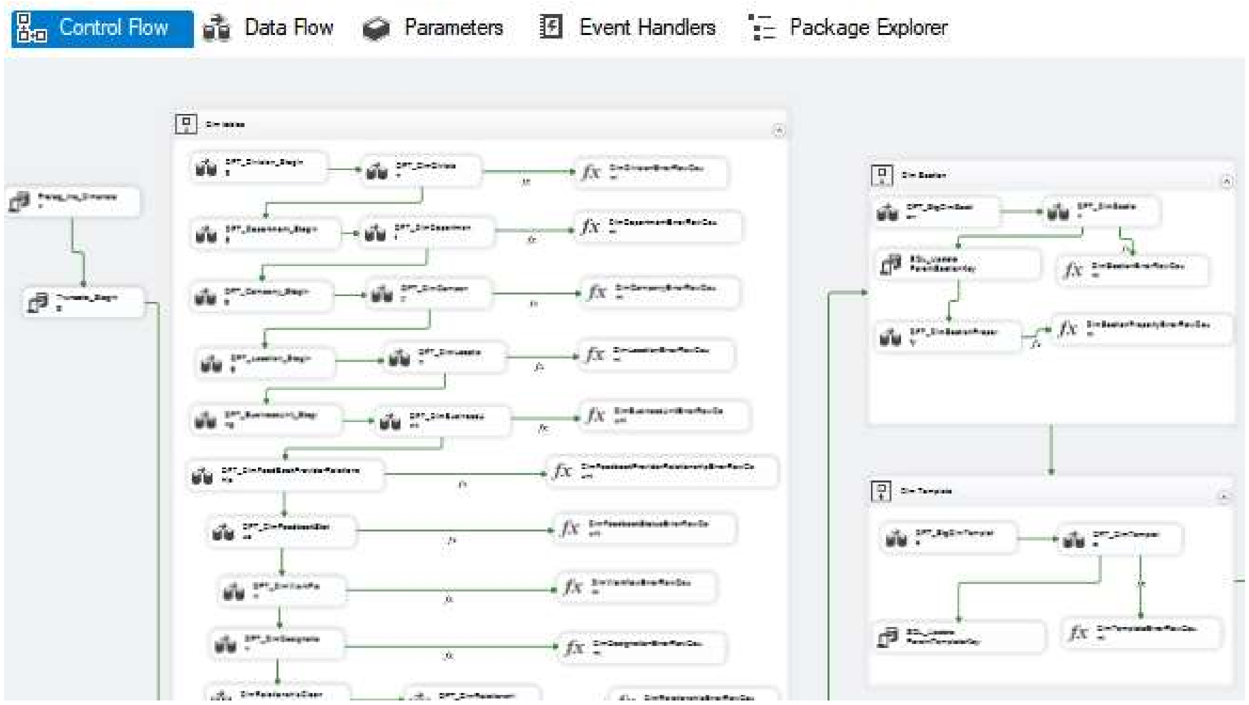
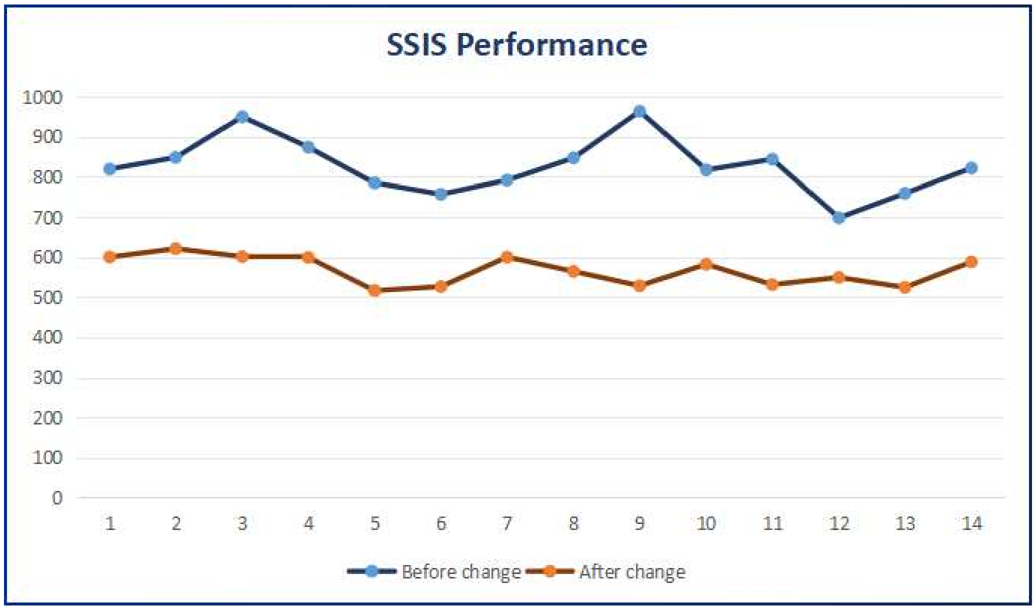


Figure 5: SQL Server Integration Service Package for Data migration task

By executing these SSIS packages, we aim to analyze performance metrics such as data throughput, execution time, memory utilization, and overall system efficiency. This



structured

Figure 6: SSIS Package Execution result Before and after the with(tablock) implementation

The approach ensures faster and more reliable data movement across heterogeneous systems, making it easier to manage large-scale data migrations in enterprise environments.

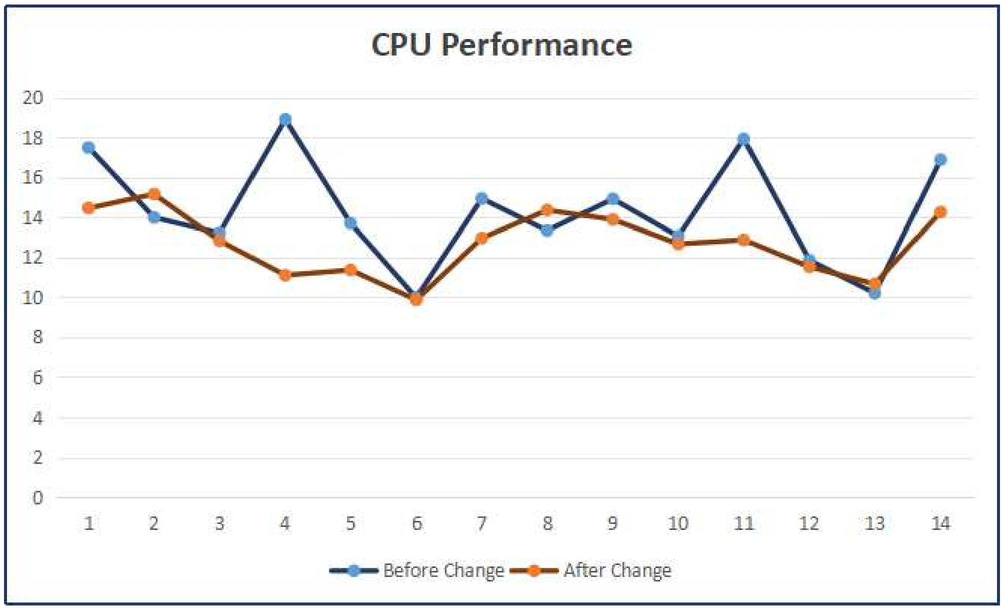


Figure 7: cloud CPU Performance impact before and after the with(tablock) Implementation

After implementing the WITH (TABLOCK) optimization in the stored procedure, we have observed a significant improvement in cloud server CPU performance. By allowing SQL Server to acquire a table-level lock instead of multiple row/page locks, the optimized query has reduced lock contention, minimized transaction overhead, and enhanced concurrency during bulk data inserts.

This improvement has led to better CPU utilization, lower processing time, and reduced resource contention, ultimately enhancing the efficiency of SSIS-based data migration workflows. The updated execution process has resulted in faster data loading, reduced query execution time, and an overall performance boost in the cloud environment.

Performance testing and monitoring indicate that CPU usage has stabilized under heavy workloads, ensuring scalability, cost-effectiveness, and reliability for large-scale data migration tasks. This enhancement strengthens the SSIS pipeline efficiency and provides a more optimized and robust cloud-based data transformation solution.

6.2. OVERALL IMPLEMENTATION RESULTS

In comparison to the previous results, The comparison of execution performance before and after the modifications reveals significant improvements in system efficiency. Before the changes, the average execution time ranged from 698.32 seconds (11.64 minutes) to 963.89 seconds (16.06 minutes) across the 15 days.

The highest execution time, recorded on 10-Jul-24, was 963.89 seconds (16.06 minutes), highlighting inefficiencies in the system. The lowest execution time during this period occurred on 13-Jul- 24, at 698.32 seconds (11.64 minutes). On average, the execution time before the modifications was approximately 817.79 seconds (13.63 minutes), underscoring the need for system optimization. After the changes were implemented, the average execution time showed a marked reduction, ranging from 516.20 seconds (8.60 minutes) to 621.09 seconds (10.35 minutes) over the next 15 days.

The best execution performance was achieved on 20-Jul-24, with an average execution time of 516.20 seconds (8.60 minutes). Even the highest execution time during this period, recorded on 17-Jul-24 at 621.09 seconds (10.35 minutes), was significantly lower than the highest per-modification time. Overall, the average execution time after the changes dropped to 561.45 seconds (9.36 minutes), reflecting a performance improvement of approximately 31.36%. The comparison highlights that the modifications effectively reduced the average execution time by 256.34 seconds (4.27 minutes), demonstrating a substantial enhancement in system performance.

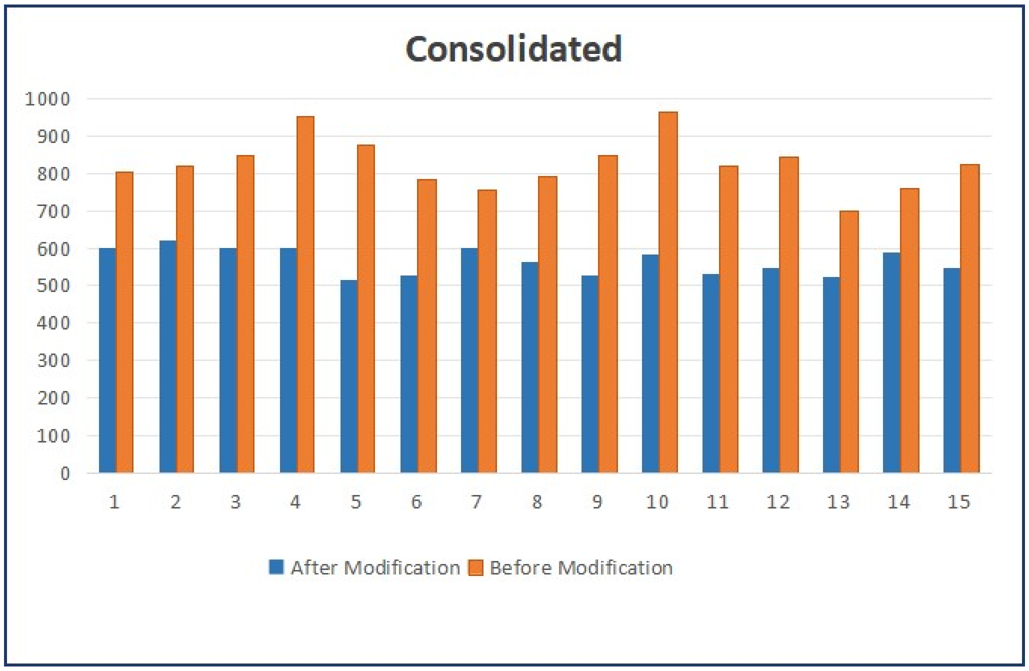


Figure 8: Consolidated Performance result in graph representation.

7. CONCLUSION

This project successfully enhances SSIS performance by implementing advanced query optimization techniques and the Bulk Transaction Lock Tuning Algorithm. By addressing key challenges such as growing data volumes, inefficient query execution, and lock contention, the proposed solution significantly improves data retrieval and loading speeds, optimizes resource utilization, and enhances system stability.

The Bulk Transaction Lock Tuning Algorithm plays a vital role in reducing lock contention, improving concurrency, and ensuring data integrity during high-volume transactions. This results in faster data processing, reduced transaction delays, and improved scalability for SSIS-based data migration workflows.

Additionally, by optimizing server utilization and reducing reliance on costly monitoring services, this approach helps minimize operational expenses while maintaining high performance and reliability. Organizations can now achieve efficient, scalable, and cost-effective data migration, enabling them to adapt to increasing data demands.

In conclusion, this project significantly improves SSIS performance, providing a robust and optimized framework for high-speed, reliable, and scalable data transfers across heterogeneous systems. The proposed solution not only enhances operational efficiency but also equips businesses with powerful tools to meet evolving data migration challenges in today's dynamic digital landscape.

REFERENCES

1. Samson, S., & Aponso, A. (2020). An Analysis on Automatic Performance Optimization in Database Management Systems. 2020 *World Conference on Computing and Communication Technologies (WCCCT)*, 6, 6–9. https://doi.org/10.1109/wccct49810.2020.9169995
2. Chen, W., & Chung, Y. (2023). An offline Profile-Guided optimization strategy for function reordering on relational databases. 2022 *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 967–972. https://doi.org/10.1109/smc53992.2023.10394026
3. Aytekin, A. (2021). Comparative analysis of the normalization techniques in the context of MCDM problems. *Decision Making Applications in Management and Engineering*, 4(2), 1–25. https://doi.org/10.31181/dmame210402001a
4. Barakhnin, V. B., Karpov, M. V., Machikina, E. P., & Musasbayev, R. R. (2024). Optimization of Database Operations in the Application for Text Corpus Analysis. *2024 20th International Asian School-Seminar on Optimization Problems of Complex Systems (OPCS)*, 1–6. https://doi.org/10.1109/opcs63516.2024.10720387
5. Zhang, H., & Rong-Li, G. (2021). Distributed HBase Cluster Storage Engine and Database Performance Optimization. *IEEE Xplore*. https://doi.org/10.1109/hpcc-dss-smartcity-dependsys53884.2021.00341
6. Abdalla, M. H., & Karabatak, M. (2020). To Review and Compare Evolutionary Algorithms in Optimization of Distributed Database Query. 2020 *8th International Symposium on Digital Forensics and Security (ISDFS)*, 1–5. https://doi.org/10.1109/isdfs49300.2020.9116418
7. Wang, L., Wang, D., & Li, W. (2021). Research on big Data query Optimization Method of power System substation equipment Condition monitoring. *2022 IEEE/IAS Industrial and Commercial Power System Asia (I&Amp;CPS Asia)*, 1479–1483. https://doi.org/10.1109/icpsasia52756.2021.9621504
8. Ding, Z. (2020). Study of Multi Ant Colony Genetic Algorithm in Query Optimization of Distributed Database. *2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM)*, 83–89. https://doi.org/10.1109/aiam50918.2020.00022

9. M. O. Lewis, B. W. Young, L. Mathiassen, A. Rai and R. J. Welke, "Business process innovation based on stakeholder perceptions", *Inf. Knowl. Syst. Manag*., vol. 6, pp. 7-27, 2007.
10. Agrawal, S., Chaudhuri, S., Kollar, L., Marathe, A., Narasayya, V., & Syamala, M. (2004). Database Tuning Advisor for Microsoft SQL Server 2005. In *Elsevier eBooks* (pp. 1110–1121). https://doi.org/10.1016/b978-012088469-8.50097-8
11. Williams, H. E., & Lane, D. (2004). *Web Database Applications with PHP & MySQL*, 2nd Edition.https://www.amazon.com/Web-Database-Applications-PHP-MySQL/dp/0596005431
12. Hastig, G. M., & Sodhi, M. S. (2019). Blockchain for Supply Chain Traceability: Business requirements and critical success factors. *Production and Operations Management*, 29(4), 935–954. https://doi.org/10.1111/poms.13147
13. Banerjee, J., & Hsiao, D. K. (1978). Performance study of a database machine in supporting relational databases. *Very Large Data Bases*, 319–329. https://dblp.uni-trier.de/db/conf/vldb/vldb78.html#BanerjeeH78
14. Christen, P. (2012). Data Matching: *Concepts and techniques for record linkage, entity resolution, and duplicate detection*. http://ci.nii.ac.jp/ncid/BB10533065
15. Schlosser, R., Kossmann, J., & Boissier, M. (2019). *Efficient scalable multi-attribute index selection using recursive strategies. 2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 1238–1249. https://doi.org/10.1109/icde.2019.00113

16. Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., & Neumann, T. (2015). How good are query optimizers, really? *Proceedings of the VLDB Endowment*, 9(3), 204–215. https://doi.org/10.14778/2850583.2850594
17. Duan, S., Thummala, V., & Babu, S. (2009). Tuning database configuration parameters with iTuned. *Proceedings of the VLDB Endowment*, 2(1), 1246–1257. https://doi.org/10.14778/1687627.1687767
18. Narayanan, D., Thereska, E., & Ailamaki, A. (2005). Continuous resource monitoring for self-predicting DBMS. *Modeling, Analysis, and Simulation on Computer and Telecommunication Systems*, 239–248. https://doi.org/10.1109/mascot.2005.21
19. Rodd, S. F., & Kulkarni, U. P. (2014). Adaptive self-tuning techniques for performance tuning of database systems: a fuzzy-based approach with tuning moderation. *Soft Computing*, 19(7), 2039–2045. https://doi.org/10.1007/s00500-014-1389-3
20. Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F. M., Rizzi, W., & Simonetto, L. (2018). Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems*, 74, 67–83. https://doi.org/10.1016/j.is.2018.01.003
21. Ganguly, S., Hasan, W., & Krishnamurthy, R. (1992). Query optimization for parallel execution. *SIGMOD '92: Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, 9–18. https://doi.org/10.1145/130283.130291