Ayden Journal of Intelligent System and Computing
Volume 1 Issue 1, February 2024
ISSN: pending...

COMPARATIVE ANALYSIS OF TRANSACTION PROCESSING: TPC BENCHMARKING FOR IN-MEMORY AND DISK-BASED DATABASES

[1]Nurul Hidayah Abdullah and [2]Siti Aishah Yusof
[1, 2]College of Information Technology, University Tenaga National, Putrajaya Campus, Malaysia

Abstract: Efficient Online Transactional Processing (OLTP) applications are essential for seamless user experiences, demanding meticulous planning and optimization of database performance. The primary goal is to minimize response times for application-specific Structured Query Language (SQL) queries. Addressing the challenges posed by concurrent user activity, often leading to congestion and resource locking, is critical for optimal performance. Drawing on insights from Kaspi, Venkatraman (2014), and TPC (2010), this study emphasizes the need for finely-tuned transactions and robust features within Relational Database Management Systems (RDBMS).The research delves into the complexities of optimizing OLTP applications, highlighting the crucial role of proactive planning and sophisticated database design amidst concurrent user interactions. Synthesizing insights from Faleiro and Abadi (2011) and established methodologies, the study provides actionable recommendations for enhancing efficiency and scalability in OLTP-based systems. Key considerations include transaction tuning, resource allocation, and the integration of advanced features within RDBMS to mitigate congestion.The study contributes to a deeper understanding of the challenges and opportunities in deploying transactional applications online. By elucidating the multifaceted dynamics of OLTP optimization, it offers valuable insights for practitioners and database administrators. The recommendations put forth in this study are crucial for organizations aiming to harness the full potential of OLTP systems in a dynamic and concurrent user-driven online environment.
Keywords: Online Transactional Processing (OLTP), Database Performance,Concurrent Users, Resource Locking, Relational Database Management Systems (RDBMS)

INTRODUCTION

Online transactional processing-based application requires comprehensive planning prior to deployment in regards to database performance and throughput. The ideal goal for the database system is to perform each transaction in shortest possible response time for application-specific structured query language (SQL) queries (Kaspi and Venkatraman, 2014; TPC, 2010). In most cases of online transaction processing (OLTP), concurrent users of the specific module of the application can cause congestion and increase resource locking (Faleiro and Abadi, 2011). With such constraints, it is evident that not only transactions have to be very tuned and carefully designed. RDBMS have to provide additional features to handle such requirement in a sophisticated manner. An in-memory database is designed to store entire data in the physical memory and update continuous changes of the data in the memory (Delaney, 2014). Tables in an in-memory database are durable and accessible using the same Transact-SQL (T-SQL) queries (Diaconu et al., 2013). In contrast to the in-memory database, disk-based database store all data on the disk, while transaction data move into main memory. As with the development of in memory database design along with no locking feature have provided enhanced performance and optimization of such systems that was not possible in disk-based database design (Diaconu et al., 2013). In a comparative study of in-memory databases, performance gain for enterprise as compared to workload was evaluated and investment is still questionable, not every type of enterprise workload can take advantage of in-memory databases (Meyer et al., 2015). In another study of mixed workload for in-memory databases, "write" performance in a mixed type of workload where OLTP and online analytical processing (OLAP) will have a drawback was analyzed. It would be important to analyse the transactional workload to get more precise in-memory implementation (Krueger et al., 2011). Benchmarking the database is performing specific tests that are close to application transactions to evaluate its performance. Response time and throughput are two factors to measure the performance. These benchmarking results can be used to measure the impact and helps in future forecasting, it allows proactive monitoring of performance bottlenecks as well. An analysis of the TPCC as Transaction Processing Performance Council Benchmark (TPC, 2010) for online transactional processing systems was performed during the course of this project to analyse the comparison between inmemory and disk-based database. In-memory database design has achieve its high performance and scalability by using very efficient latch free data structures, multi-versioning, a new optimistic concurrency control scheme, and by compiling T-SQL stored procedure into efficient machine code (Diaconu et al., 2013). The main hypothesis of this study is to outline the performance differences between in-memory and disk-based database in conjunction of concurrent users and parallelism. These two factors will cover throughput and concurrency aspect of application workload. This comparison measured with industry standard benchmarking specification which covers all the aspects of transactional consistency and concurrency like production applications. The focus of the study is to analyse the comparison and review the possible improvement area of read/write performance of disk-based database as well as the inmemory database. Database schema design is as per TPC-C specifications, initial database schema and data size will be the same for both type of database. There were two main test cases in observation during the course of the project. The first test case covers comparison of single and concurrent users for the inmemory and disk-based database and the second test case includes query parallelism setting.

RELATED WORK

A detailed comparative analysis was provided in an article (Saikia et al., 2015). This analysis covered performance measurement of a specific application system where backend was MySQL and SQL Server, respectively. Different types of queries that include SELECT, INSERT, UPDATE and DELETE were performed and response time was examined; based on query response time, performance was analyzed. MySQL and SQL Server did not use in-memory feature during the experiment. Another discussion was reported by Raja et al. (2006) about performance comparison between FastDB and SQL Server. In this article, FastDB was used in-memory feature whereby disk-based database was hosted on SQL Server. Based on TPC-C benchmarking, performance was evaluated for queries. This study compared two different relational database management system (RDBMS) with a different technique to handle similar tasks. In-memory databases, performance was evaluated in a study (Kabakus et al., 2016) of open source nonconventional database systems. In open source database management systems, atomicity, consistency, isolation, durability (ACID) consistency is reduced in order to provide high-performance transactional throughput. During the evaluation of few open source database management systems, each standout in one type of transactions. SQL based databases provide complete consistency that cannot be replaced by NoSQL databases. A comprehensive study (Meyer et al., 2015) elaborated the commercial aspect of in-memory databases. The study provides an analysis of different workload and there technological requirement whereby in-memory is suitable or disk-based. The study evaluated that in-memory databases are not always faster as compared to disk based database depending on number of users and workload characteristics. The analysis of same RDBMS from in-memory and disk-based characteristics was not analysed in any of these studies using the same benchmarking specifications and workload. Referring to Table 1, the first three articles discussed about different database technologies and their differences. In articles 1 and 2, in-memory feature was discussed and compared with totally different database technology without in-memory feature. Article 4 measured the differences between disk-based and inmemory database with enterprise applications whereby OLTP and OLAP transactions were mixed and TPC-C benchmark was not used as well.

Table 1. Related work.

No.	**Article**	**Cross database technology**	**Comparison with in-memory database**
1	Comparative performance analysis of MySQL and SQL Server Relational Database Management Systems in Windows Environment (Saikia et al., 2015)	Yes, MySQL and SQL Server	None of databases was measured with inmemory feature
2	A comparative study of Main Memory Databases and Disk-Resident Databases (Raja et al., 2006)	Yes, FastDB and SQL Server	Only FastDB used in-memory feature, SQL Server used disk-based database
3	A performance evaluation of in-memory databases (Kabakus et al., 2016)	Yes, SQL and NoSQL	In-memory feature was compared with SQL and NoSQL based databases
4	Assessing the suitability of in-memory databases in an enterprise context (Meyer et al., 2015)	No, however used linux operating system	In-memory feature was compared with different type of workload other than TPC

Performance evaluation

Performance evaluation of database requires comprehensive defined tests to measure two main areas, throughput and response time (Kaspi and Venkatraman, 2014; TPC, 2010). In terms of comparison of disk-based and in-memory table design, it is important to have either same hardware or identical servers with same internal configuration and parameters. Statistical information on performance comparison test results will provide important information for making decisions for proactive database scalability. This evaluation helps in continuous measurement of database landscape and pinpoint capacity growth and changes affected by new version or patches. Performance evaluation is an ongoing process, which is informative as compared to benchmark while configuration parameters have specific changes. Different organizations have different workloads; even within a particular organization, these workloads represent various statistics to measure and benchmark for future cross verifications. In general, the following are the most popular database workload types (Kaspi and Venkatraman, 2014; Elnaffar et al., 2002):

(1) OLTP: Online Transaction Processing (2) OLAP: Online Analytical Processing/DSS: Decision Support System. Results of performance evaluation will provide important comparative aspects of different types of

objects, in the present case in-memory and disk-based tables. It allows proactive understanding, that which type of object will be beneficial and at what type of workload, so bottlenecks can be avoided.
There are different ways of performance evaluation for databases, one as defined by software vendors or organizations and usually pre-evaluated based on standard parameters and workload; second is as defined by TPC (Elnaffar et al., 2002; TPC, 2010).

Transaction processing performance council (TPC)

The objective of TPC benchmarks is to offer relevant objective performance data to industry users. To accomplish that purpose, TPC benchmark specifications require that benchmark tests be implemented with systems, products, technologies, and pricing (TPC, 1994, 2010). TPC Benchmark C (TPC-C) is used for OLTP workload. It combines read and update transactions that are more specific to OLTP application. The performance metric statistical report generated by TPC-C is a "business throughput" that calculates the number of processed orders per minute, concurrent orders processing simulated based on response time. The results of performance metric represents in transactions-per-minute-C (tpmC) (TPC, 2010).
The properties of the TPC-C as reported by the TPC Benchmark C Standard Specification Revision 5.11 are given as follows (TPC, 2010):

(1) Is the implementation commonly available including documentation and vendor supported?

(2) Does the implementation have substantial constraints on its use or applicability that confines its use beyond TPC benchmarks?

(3) Is there any portion or full implementation poorly incorporated into the larger product?

(4) Does the implementation take unusual benefits of the imperfect nature of TPC benchmarks (e.g., transactions, transaction combination, transaction concurrency and/or contention, and transaction isolation) in a way that would not be normally applicable to the environment the benchmark represents?

(5) Is the use of the implementation discouraged by the vendor (This comprises failing to stimulate the implementation in a way comparable to other products and technologies)?

(6) Does the implementation need complexity on the part of the system administrator, programmer or end-user?

EXPERIMENTAL SETUP

Microsoft SQL Server 2014, Enterprise Edition, 64 bit on Microsoft Windows Server 2012 64 bit (Microsoft, Server 2017), installed on a machine with a core 2 quad CPUs at 1.80 GHz each, 12 GB of physical memory and 500 GB of the hard drive. The way of dealing with the problem presented in this project was to investigate the workload specified in TPC-C specifications and compare the performance of disk-based and in-memory design with its default settings. Performance monitoring tool collected the information related to workload execution and later comparative analysis was performed to understand the differences. The analytical idea includes fine-tuning the indexes and changing the query parallelism to analyse the differences as well. Two databases with the names as "TPCC_Disk_5GB" and "TPCC_Memory_5GB" were created with the TPC-C workload. These databases will be referred to as: D5 = TPCC_Disk_5GB and M5 = TPCC_Memory_5GB. Database size was selected to observe the significant difference in query response time and database size can fit in main memory as well. The process of generating the data to build workload was done by "HammerDB" open source tool which loads the necessary data for specific size and type of databases. HammerDB open source software was used to generate two different databases. The default configurations were left for the two databases and SQL Server along with the operating system. Exactly 10000 times stored procedures were executed using HammerDB per user to generate reasonable transactional stress. The Microsoft SQL Server Profiler was setup to capture the duration of each stored procedure and queries within it, CPU time to process and disk reads and writes for each transaction.

Database design of TPC-C specification

TPC-C specification provides a database consisting of nine tables; the cardinalities depend upon the size of database and data generated for a specific size. Table 2 shows the table level number of rows that are used during the project for in-memory and diskbased databases.

Table 2. Table cardinalities.

Table name	Database size (5 GB) row count
Customer	1,200,000
District	400
History	1,200,000
Item	100,000
new_order	360,000
order_line	11,997,485
Orders	1,200,000
Stock	4,000,000
Warehouse	40

Measurements

The performance measurements were collected with "SQL Server Profiler" software built-in SQL Server which was used to capture SQL events specifically stored procedure and SQL queries within stored procedures.

The performance measurements were:

(1) The stored procedure or query response (Duration in microsecond) taken to execute a single execution.

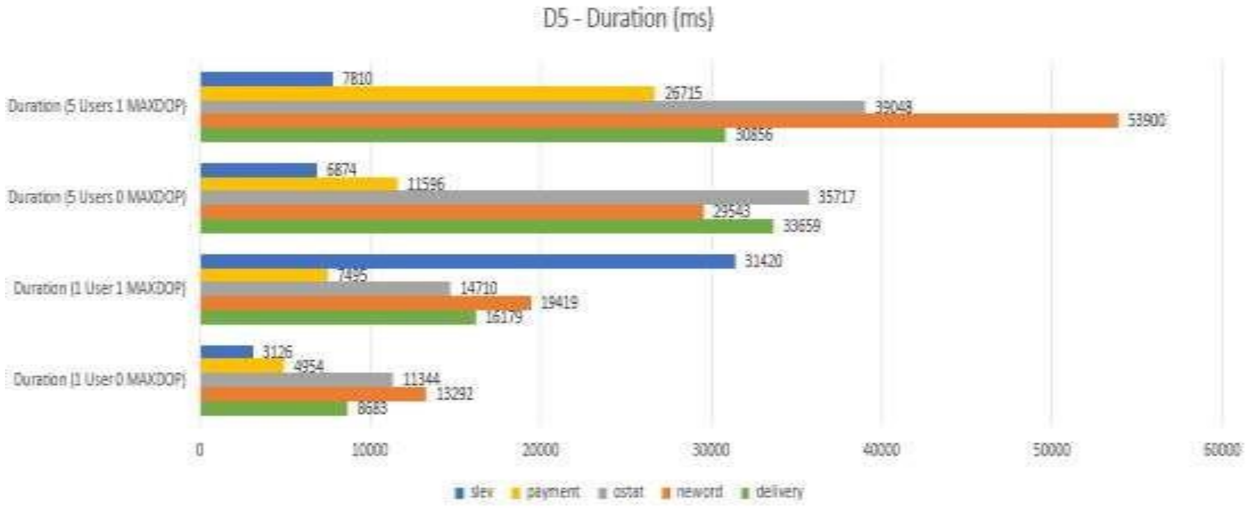
(2) The CPU time (in millisecond), the aggregated time CPU spent on the processing of specific stored procedure or query.

(3) Disk read, that provides number of reads of 8K pages from either the cache or disk.

(4) Disk write, that provides number of writes of 8K pages to either the cache or disk.

Performance comparison

Once data generation process is complete, the first performance test was performed with default configurations on D5 database and similar for M5 as well. In the second step of concurrent users, analysis will take place where additional users will be executing the transactions. First comparison test will be with default configurational parameters against each type of database. There was performance monitoring tool “Profiler” (Microsoft, Profiler, 2017) configured to capture statistics. These performance statistics will cover different aspects of utilization to understand the workload and its throughput. Once performance test with default settings is complete, analysis of performance monitoring statistics took place and while applying additional stress using concurrent users to analyze the differences. In the third attempt of performance analysis, maximum degree of parallelism setting (Fritchey, 2012) for Apress, Fritchey (2012) for Simple Talk and (Nevarez, 2010) was changed to 1 from 0. This setting restricts the parallel execution plan to use 1 CPU core whereby the value of 0 which is the default and used for all available processors. During the database performance comparison tests, each user session have executed 10,000 random and sequential mixed transactions where each transaction duration was measured. While comprising the statistics between in-memory and disk-based transactions, „average‟ duration of specific transaction was collected for comparison. The main reason of choosing the „Average" based analysis is disk-based transactions which use locks for data concurrency which do not exist in in-memory database transactions (Diaconu et al., 2013). As a result, transaction duration can vary based on the number of users and parallelism of transactions for disk-based database. In order to compare the differences between these two types of databases „average" based analysis is the most appropriate.



Graph 1. Statistics of average duration against D5.

RESULTS AND DISCUSSION

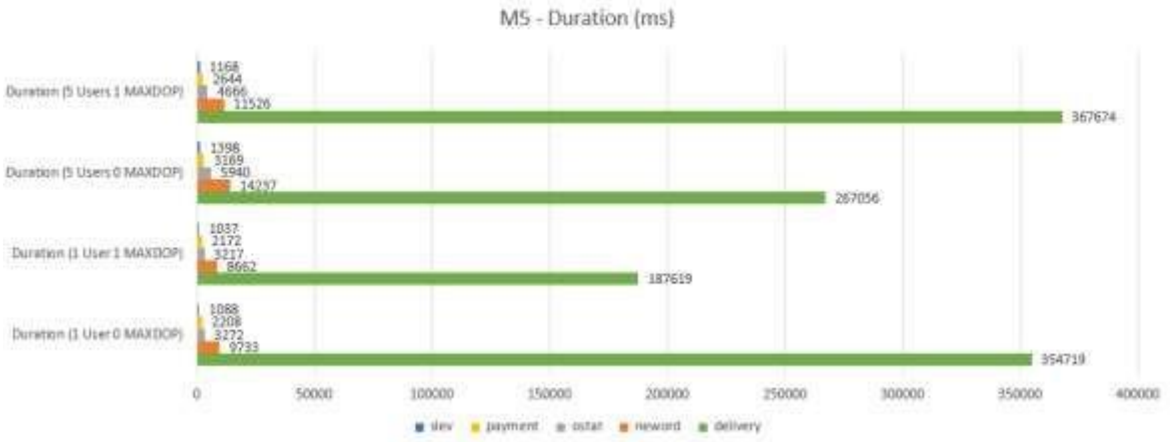
The objective of this project was to investigate the TPC-C benchmark suite for Online Transactional Processing systems; the test cases with query parallelism and concurrent users were evaluated for performance comparison. The two techniques were used to gauge the performance between disk-based and in-memory databases. The use of multiple concurrent users to simulate the real-time transactional load and changes in query parallel processing settings. Read and write response time by the transactions were significantly reduced. However, delete queries have shown additional overhead to transactions response time that has added stress on CPU for in-memory TPCC table model.

Test case 1: Concurrent users

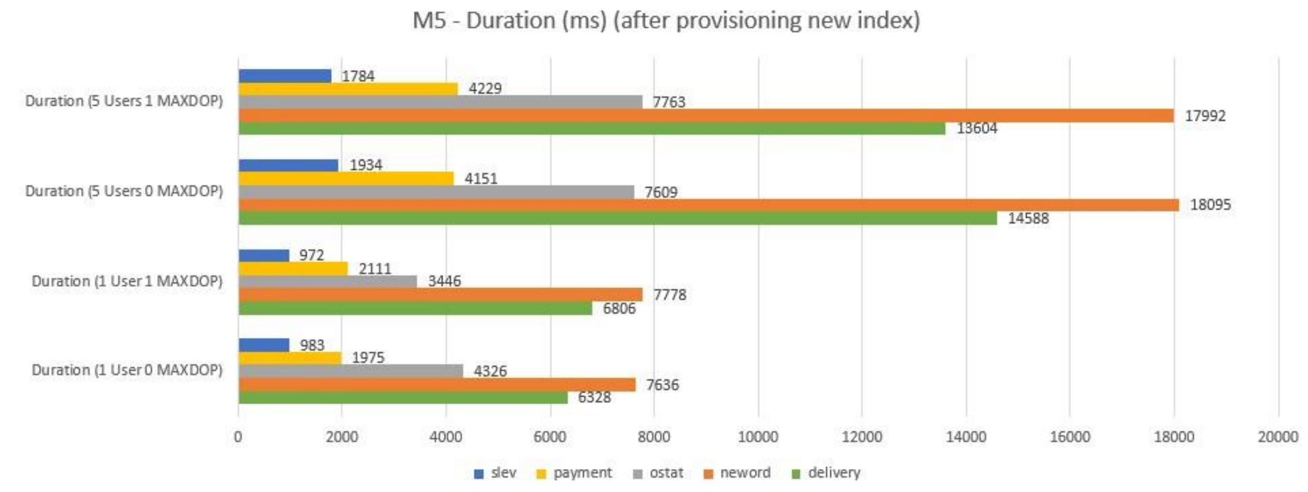
Five concurrent users apply additional load on the database as compared to a single user and replicate real-world scenario as well. The combination of multiuser and multicore setting provide comprehensive statistics to review the type of transactions that can be benefited with a specific setting. For the disk-based database, concurrent user"s transactions have increased the response time of each stored procedure as SQL Server optimizer uses lock escalation strategy for disk-based databases. In one scenario, changing the maximum degree of parallelism (MAXDOP) setting from 0 to 1 has benefited "delivery" stored procedure. In case of an in-memory database, concurrent users have not increased the response time of stored procedures. However, very slight increase in response time is noticeable which is not comparative with diskbased database where the proportion is many times more than in-memory response time. While comparing Graphs 1 to 3 and respective tables, it is clear that in-memory transactional throughput has outperformed the disk-based transactional throughput. However, MAXDOP setting is an important configuration which requires thorough testing of specific workload before provisioning it to the production environment.

Test case 2: Parallel query processing

By default, SQL Server utilizes all available processors to utilize multiprocessor architecture. The aim of changing parallelism setting is to compare the results acquired with available four CPUs as compared to one CPU, so differences of query response time can be measured. The first measurement was for the disk-based database (D5) whereby the average duration of stored procedures was collected. Based on statistics, limiting the CPU to one has caused slowness for all stored procedures in single user transactions. It is the same observation while in 5 concurrent users, except for "delivery" of stored procedure that has benefited slightly (Graph 1 and Table 3).
The second measurement was for the in-memory based database (M5), based on the average duration of stored procedure that was collected, four out of five stored procedures significantly improved the response time. However, "delivery" of stored procedure was even slower than the disk-based database (Graph 2 and Table



Graph 2. Statistics of average duration against M5.



Graph 3. Statistics of average duration against M5 (with additional index).

Table 3. Statistics of average duration against D5.

Stored procedure	Duration (1 User, 0 MAXDOP)	Duration (1 User, 1 MAXDOP)	Duration (5 Users, 0 MAXDOP)	Duration (5 Users, 1 MAXDOP)
delivery	8683	16179	33659	30856
neword	13292	19419	29543	53900

ostat	11344	14710	35717	39048
payment	4954	7495	11596	26715
slev	3126	31420	6874	7810

4).

After provisioning, new index on “new_order” table, the response time of “delivery” of stored procedure was significantly reduced; for example, 6328 ms from 354719 ms in the test case of 1 user and MAXDOP 0 (Graph 3 and Table 5).

Table 4. Statistics of average duration against M5.

Stored procedure	**Duration (1 User, 0 MAXDOP)**	**Duration (1 User, 1 MAXDOP)**	**Duration (5 Users, 0 MAXDOP)**	**Duration (5 Users, 1 MAXDOP)**
delivery	354719	187619	267056	367674
neword	9733	8662	14237	11526
ostat	3272	3217	5940	4666
payment	2208	2172	3169	2644
slev	1088	1037	1398	1168

Table 5. Statistics of average duration against M5 (with additional index).

Stored procedure	**Duration (1 User 0 MAXDOP)**	**Duration (1 User 1 MAXDOP)**	**Duration (5 Users 0 MAXDOP)**	**Duration (5 Users 1 MAXDOP)**
delivery	6328	6806	14588	13604
neword	7636	7778	18095	17992

ostat	4326	3446	7609	7763
payment	1975	2111	4151	4229
slev	983	972	1934	1784

Table 6. P-value of individual transactions in comparison of both type of databases

Stored procedure	User 1, MAXDOP 0	User 1, MAXDOP 1	User 5, MAXDOP 0	User 5, MAXDOP 1
delivery	8.9301 e-12	4.00838 e-13	8.44983 e-32	8.94195 e-11
neword	2.99396 e-22	3.57237 e-07	1.8841 e-167	8.81321 e-10
ostat	1.77215 e-07	1.33781 e-20	1.32242 e-05	2.77474 e-82
payment	9.09524 e-25	2.51412 e-32	1.6116 e-111	8.7204 e-241
slev	1.903 e-140	0.160209722	2.0144 e-18	2.46986 e-13

Analysis with T-Test

This study demonstrated the results based on average analysis that in-memory database performance is very significant as compared to disk-based. In order to validate the results, we have chosen to analyze the results with T-Test statistical test. Transaction of each database was randomly generated where only number of transaction per user was constant to 10,000 so independent unequal two-sample variance of T-Test was used to measure the probability of differences between both type of databases. Table 6 shows that the p-value between disk-based and in-memory is significantly lower (e.g. p-value of delivery stored procedure for one user with MAXDOP setting to 0 was 0.000000000008930).

Conclusion

It is important to understand the application of transactional activities before provisioning it to in-memory database environment; the index strategy in disk-based and in-memory is different and required careful testing and thorough review. It is an important point especially while in migration. The experiments conducted in this project have proved that like-to-like migration will cause severe performance bottlenecks for the application that will be using the specific database. Parallel query processing has to examine carefully before implementation. The experiment suggests that lightweight queries can take advantage of sequential execution and might run faster as compared to parallel.

Due to the extensive and permanent usage of memory by the in-memory tables, database required sufficient physical memory and other resources have to be planned. The in-memory design increases the recovery time as well.

RECOMMENDATION FOR FUTURE WORK

Future work in the progression of this project topic can be:

(1) A comparison of data warehouse workload with the disk-based and in-memory design.

(2) A comparison study with other relational database management systems to review the different transactional enhancement, especially indexes.

(3) Impact of disaster recovery feature, for example, synchronized database mirroring in conjunction with this study.

(4) A study to review in different hardware, especially SAN and clustered environment.

CONFLICT OF INTERESTS

The author has not declared any conflict of interests.

REFERENCES

Delaney K (2014). SQL Server Internals: In-memory OLTP. Simple Talk Publishing https://www.red-gate.com/simple-talk/books/sql-books/sqlserver-internals-in-memory-oltp/

Diaconu C, Freedman C, Ismert E, Larson P, Mittal P, Stonecipher R, Verma N, Zwilling M (2013). Hekaton: SQL Server‟s Memory-Optimized OLTP Engine. SIGMOD pp. 1243-1254

Elnaffar S, Martin p, Horman R (2002). Automatically classifying database workloads. ACM pp. 622-624.

Faleiro J, Abadi D (2011). Rethinking serializable multiversion concurrency control. Proceedings of the VLDB Endowment 8(11):1190-1201.

Fritchey G (2012). SQL Server 2012 Query Performance Tuning - Third Edition. Apress, Berkeley, CA. pp. 15-57.

Fritchey G (2012). SQL Server Execution Plans - Second Edition. Simple Talk Publishing. https://www.red-gate.com/library/sql-serverexecution-plans-2nd-edition

Kabakus A, Kara R (2016). A performance evaluation of in-memory databases. Journal of King Saud University – Computer and Information Sciences pp. 520-525.

Kaspi S, Venkatraman S (2014). Performance Analysis of Concurrency Control Mechanisms for OLTP Databases. International Journal of Industrial Engineering and Technology 4(4):313-318.

Krueger J, Grund M, Boissier M, Zeier A, Plattner H (2011). "Data Structures for Mixed Workloads in In-Memory Databases", IEEE. https://ieeexplore.ieee.org/document/5711090/

Meyer R, Banova V, Danciu A, Prutscher D, Krcmar H (2015). Assessing the suitability of in-memory databases in an enterprise context. IEEE pp. 78-89.

Microsoft, Profiler (2017). SQL Server Profiler. Retrieved October 6, 2017, From https://docs.microsoft.com/en-us/sql/tools/sql-serverprofiler/sql-server-profiler

Microsoft, Server (2017). What's New in Windows Server. Retrieved October 6, 2017, From https://technet.microsoft.com/enus/library/dn250019(v=ws.11).aspx

Nevarez B (2010). Inside the SQL Server Query Optimizer. Simple Talk Publishing.

Raja F, Rahgozar M, Razavi N, Siadaty M (2006). A Comparative Study of Main Memory Databases and Disk-Resident Databases". World Academy of Science, Engineering and Technology 14:128-131.

Saikia A, Dolma S, Mary R (2015). Comparative Performance Analysis of MySQL and SQL Server Relational Database Management Systems in Windows Environment. IJARCCE pp. 160-164.

TPC (2010). TPC Benchmark C Standard Specification Revision 5.11. URL: http://www. tpc. org/tpcc.

TPC. (1994). TPC Benchmark B, A Standard Specification, Revision 2.0.