Microsoft

Microsoft SQL Server – The Case for Single Node Systems Supporting Large Scale Data Analytics

Summary: Cluster-based configurations have emerged as a widely adopted paradigm for storing and processing prodigious amounts of data at economies of scale. While multi-node configurations remain attractive to support the growing demands of elasticity, and data and resource scalability, single node systems have also evolved in their processing power with up to hundreds of cores and advanced storage and memory subsystems at a compelling total cost of ownership (TCO). Consequently, single-node relational database platforms should not be overlooked for large scale analytics.

Intel
Authors: Dippy Aggarwal, Chris L. Elford, Shreyas Shekhar, Avaneesh Shetty
Acknowledgements: Vaishali Paliwal, Mahmut Aktasoglu

Microsoft
Authors: Joe Sack, Vassilis Papadimos
Acknowledgements: Jamie Reding, Sadashivan Krishnamurthy

Published: April 2021
Version 1.1.0

Contents

Forward

In this white paper, we demonstrate the capabilities of single-node relational databases by executing an analytical benchmark derived from TPC-DS using an enterprise-grade relational database engine (Microsoft® SQL Server 2019). The benchmark was executed on a single-node system powered by Intel® Xeon™ Platinum 8280L product family and enterprise-class Intel® SSDs. The past few years have shown an increased interest in the industry in embracing TPC-DS derivatives to demonstrate their product's capabilities and performance. However, the existing studies fall in one of the following two categories:

(1) they are either based on a subset of TPC-DS queries which are cherry-picked based on how well they showcase one's product, or
(2) they are executed on a cluster-based configuration.

To the best of our knowledge, this paper is the industry's first offering that demonstrates a functional view of executing all 99 TPC-DS queries on a single-node configuration. We present results from our initial experiments using the 1 TB and 3 TB data sets which serve as meaningful scale sizes, popular for small to mid-size businesses. The queries are derived from the industry-standard benchmark TPC-DS, and the results presented in this paper are based on a sequential run of all 99 read-only ad-hoc queries (i.e., power run) over each of the two datasets.

In particular, the results in this paper include:

- A description of the system configuration used for running the power run experiments and an insight into the data load process. We have employed the clustered columnstore (CCI) index feature of Microsoft® SQL Server 2019 to build the database. Clustered columnstore indexes are known to offer significant gains in query performance and storage for data warehousing workloads [5]. Given the large number of queries, we present our results by clustering queries around runtimes and their resource utilization behavior.
- While the goal of this paper is to offer a capability demonstration of relational databases and the power of single-node configurations for running all 99 TPC-DS queries, we also provide a holistic overview of the benchmark. To this end, the paper also provides a conceptual overview of multi-tenant use cases (described as "throughput run" by TPC-DS spec) and data maintenance phases of the benchmark followed by conclusions and next steps.

The intended audience for this paper includes database performance analysts, IT executives, solution architects, and infrastructure planners. The results presented in the paper should not be compared to any of the published TPC results for the two scale factors mentioned above.

TPC-DS - Introduction

TPC-DS belongs to a class of decision support, analytical benchmarks provided by a benchmark standardization organization, Transaction Processing Council (TPC). TPC is a consortium with experts from industry and academia, responsible for defining, maintaining, and auditing database benchmarks. The TPC-DS benchmark emulates a data warehouse, which represents a specialized database that models data in an efficient manner for analytical queries. It reflects the business use case of a fictitious retail product supplier which manages its sales and returns through three distribution channels – store, web, and catalog.

As the industry's premiere decision support benchmark, the workload exhibits the following characteristics.

- stores and analyzes large volumes of data.
- stresses all major components including compute, memory, and I/O subsystems.
- supports a data modeling approach that is favorable to analytical queries vs. transaction processing operational systems.
- consists of queries representing real-world questions for the modelled business use-case.
- maintains synchronization with the data from operational systems through data maintenance functions, and
- defines a performance metric that considers the time it takes to load the data in the database from flat files, a sequential run of all benchmark queries, throughput as measured by concurrent execution of multiple queries, and a data maintenance phase.

Schema overview

While TPC-DS is gaining popularity in the recent years as evident by industry white papers and TPC publications [6], TPC-H has been the benchmark of choice. It has been widely adopted by both industry and academic research for over a decade [4, 8, 9]. Both the benchmarks, TPC-H and TPC-DS, model the business use case of a retail product supplier. Some of their differentiating factors are outlined below. While the TPC-H schema (specification: 2.18.0) consists of 8 tables with 16 columns for the largest table (lineitem), the TPC-DS schema (specification: 2.13.0) includes 26 tables with its largest table (store_sales) consisting of 23 columns. Furthermore, TPC-DS uses snowflake schema which is a combination of 3NF and star schema [1] and is significantly more complex than other TPC workloads such as TPC-H (see [1], [3] for more details comparing the two benchmarks and highlighting the additional value that TPC-DS offers compared to the TPC-H benchmark).

Workload Phases

Executing the TPC-DS benchmark entails the following steps:

- **Raw data generation:** This involves generating raw data in the form of flat files for a given scale factor. The scale factor corresponds to the approximate size of raw data in gigabytes. This step of raw data generation is accomplished using a tool provided in the official benchmark kit [2]. Compared to TPC-H, TPC-DS reflects a skewed data distribution which is more representative of real-world content [1, 3].
- **Data load phase:** This involves creating the database schema including tables, indexes, constraints (primary/foreign keys) if any, and populating the raw data from flat files into the tables.

- **Power run:** This includes executing all the 99 benchmark queries in a specific order prescribed by the benchmark specification [2].
- **Throughput run:** This extends the power run phase by executing multiple streams of queries concurrently with each stream including all the 99 queries. The simultaneous execution of these queries is aimed at simulating multiple users executing queries against the database concurrently.
- **Data maintenance:** Also referred to as data refresh step, this includes executing a set of inserts, delete, and update operations to mimic refresh operations that are an integral part of a data warehouse environment. It covers not only the time to complete these data modification operations but also the time it takes to update any auxiliary data structures such as indexes that are often created to improve query times.

Except for the first step of raw data generation, all other phases are timed and contribute to the benchmark metric score (in different ratios) defined by TPC-DS benchmark specifications. This is another factor that differentiates TPC-DS from TPC-H where data load is not accounted in the overall benchmark score. We believe that incorporating a certain percentage of the time it takes to load data from flat files into the database can serve as a valuable addition for customers.

The TPC-DS metric is computed as the geometric mean including a fraction of the time taken for data load, and elapsed times for power, throughput runs, and data maintenance phases. In this paper, we present initial results from the power run portion of the benchmark.

Setting up our TPC-DS Derivative

Building the database

We have chosen two scale factors, 1 TB and 3 TB, to test our setup. The database is constructed based on the TPC-DS schema described in the TPC-DS specification [2] and uses the power of clustered columnstore technology offered by Microsoft SQL Server. Clustered columnstore indexes are known to excel at processing data warehousing and analytical workloads by offering up to 100x performance gains in query processing and up to up to 10x data compression compared to traditional row store indexes [5]. In addition to employing clustered columnstore indexes, we leverage the power of multi-threaded execution to load data into our databases.

The data is loaded from flat files in a multi-threaded manner resulting in effective CPU and I/O utilization. To maximize CPU and IO parallelism, we use multiple data files for the database and raw data for each of the tables are split across multiple flat files. At the hardware level, our I/O subsystem is configured in a manner that supports high read/write throughput. Detailed configuration settings are presented in Table 1.

Though we leverage the 99 queries provided by TPC-DS benchmark, it should be noted that our setup differs from an official TPC-DS benchmark in terms of storage redundancy requirements, detailed cost analysis of the hardware used usage of (non-standard) Microsoft SQL Server trace flags and has not been audited by a TPC certified auditor. Thus, the results presented in the paper should not be compared to any of the published TPC results for the two scale factors mentioned above.

Technologies

In 2016, we released a whitepaper detailing the value proposition of Microsoft SQL Server 2016 using a derivative of the TPC-H benchmark [4]. The same SQL Server features are also relevant to TPC-DS derivatives and are described below.

Microsoft® SQL Server 2019 technologies

Columnstore and Data compression

Large data warehousing databases typically benefit by using columnstore indexes. Columnstore indexes use compression-friendly column-based data storage which, in addition to reducing storage requirements, improves query performance by using a smaller in-memory footprint.

Additional compression can be enabled by configuring archival compression. Archival compression will further reduce the data size but may reduce query performance due to the extra CPU resources needed to retrieve archival compressed data.

Batch-mode Processing

SQL Server uses batch processing mode to process many rows at a time rather than doing computations row by row. This is particularly attractive for analytical workloads such as TPC-DS, which consists of tables with millions or billions of rows. At the hardware level, the performance of batch mode operators is optimized by leveraging Single Instruction, Multiple Data (SIMD) vector-processing CPU instructions in the Intel® architectures. Each column within a batch is stored as a vector in a separate area of memory, so batch mode processing is vector-based. Batch mode processing also uses algorithms that are optimized for the multi-core CPUs and increased memory throughput that are found on modern hardware platforms. Batch mode execution is tightly integrated with and optimized around the SQL Server columnstore storage format[1]. Batch mode processing will operate on the compressed data when possible and eliminates the exchange operator used by row mode execution. When a query is executed in batch mode, and accesses data in columnstore indexes, the execution tree operators and child operators read multiple rows together in column segments. SQL Server reads only the columns required for the result, as referenced by a SELECT statement, JOIN predicate, or filter predicate. The overall result is better parallelism and faster performance.

System Technologies

Intel® Xeon™ Processor Based Platforms

For our analysis, we used a 2-socket platform with Intel® Xeon™ Platinum 8280L processors. The configuration provides us with 28 physical cores per processor and 56 logical processors (hyperthreading enabled). Infrastructures built on the Intel® Xeon™ Platinum 8280L processor can deliver real-time analytics services and open new data-driven business opportunities. Designed for the most mission-critical workloads and the always-on enterprise, the processor provides improved memory subsystem performance with memory speeds up to 2933 MT/s and combines large memory capacities with high performance, reliability, and virtualization capabilities to keep data centers supplying business advantage

[1] Batch Mode on Rowstore is also supported by SQL Server. (Intelligent query processing - SQL Server | Microsoft Docs https://docs.microsoft.com/en-us/sql/relational-databases/performance/intelligent-query-processing?view=sql-server-2017)

without interruption. In our configuration, the system has fully populated DDR4 memory of 3 TB with a memory frequency of 2666 MT/s. Intel® Xeon™ Platinum processors also bring a rearchitected cache hierarchy designed for server workloads along with the implementation of features like AVX-512, support for Intel® Optane® persistent memory and Security mitigations for L1 Terminal Fault and Meltdown.

Intel® Data Center SSDs

The storage subsystem includes 4 Intel SSD DC P4610 NVME devices along with a single Intel SSD DC 3500 used for logging. Intel® SSDs for the data center are optimized for performance, reliability, and endurance. By combining the Intel® Data Center family Non-Volatile Memory Express (NVMe) along with Intel® CPUs, chipsets, firmware, and drivers, we built a seamless system; enabling large amounts of data to be transferred to processors quickly to eliminate I/O bottlenecks.

System Configuration

CPU		
Processor	Intel® Xeon™ Platinum 8280L	
# Sockets/Cores/Threads	2/56/112	
Last Level Cache	38.5MiB	
Base/Turbo Frequency	2.7GHz/3.4GHz	
TDP	205W	
QPI	3 links, 10.4GT/s	
Instruction Set Extensions	Intel® SSE4.2, Intel® AVX, Intel® AVX2, Intel® AVX-512	
PLATFORM		
Chipset	Intel® C628 Chipset	
# PCI Express® (PCIE®) ports	8	
STORAGE		
Data + Tempdb + Backup	Intel® SSD DC P4610 (NVMe)	x4
Log	Intel® SSD DC S3500 (SSD)	x1
Operating System	Intel® SSD DC S710 (SSD)	x1
MEMORY		
Memory Type	DDR4	
DIMM Size and Type	128GB RDIMM, dual rank	
Memory Frequency (MHz)	2666	

DIMMs/Channel	2
System Memory Capacity (GB)	3072
SOFTWARE	
Database Software	Microsoft® SQL Server 2019 Enterprise RTM 64-bit Edition
OS Distribution	Microsoft® Windows Server 2019, Version 1809
SECURITY MITIGATION	
Mitigations Enabled (Software)	SpectreV2, SSBD (Speculative Storage Bypass Disable)
Hardware Mitigations	L1 Terminal Fault, Meltdown
DATABASE SPECIFICS	
Scale Factor	1 TB and 3 TB
Database files	32

Table 1. System Configuration for experiments.

Results & Analysis

While we discuss the data load process for our 1 TB and 3 TB setups, the main goal of the paper is to illustrate the capabilities of Microsoft SQL Server 2019 to run all 99 queries (power run phase) of the TPC-DS benchmark on a single node system. As mentioned previously, this is the first functional run of all the power queries for the TPC-DS benchmark on a single-node configuration.

The graph in Figure 1 depicts the runtimes of 99 queries using logarithmic scale for readability. Each bar represents one of the 99 queries, sorted in increasing order of log of their completion time. The results in Figure 1 include the initial set of optimizations outlined in the query optimization effort section.

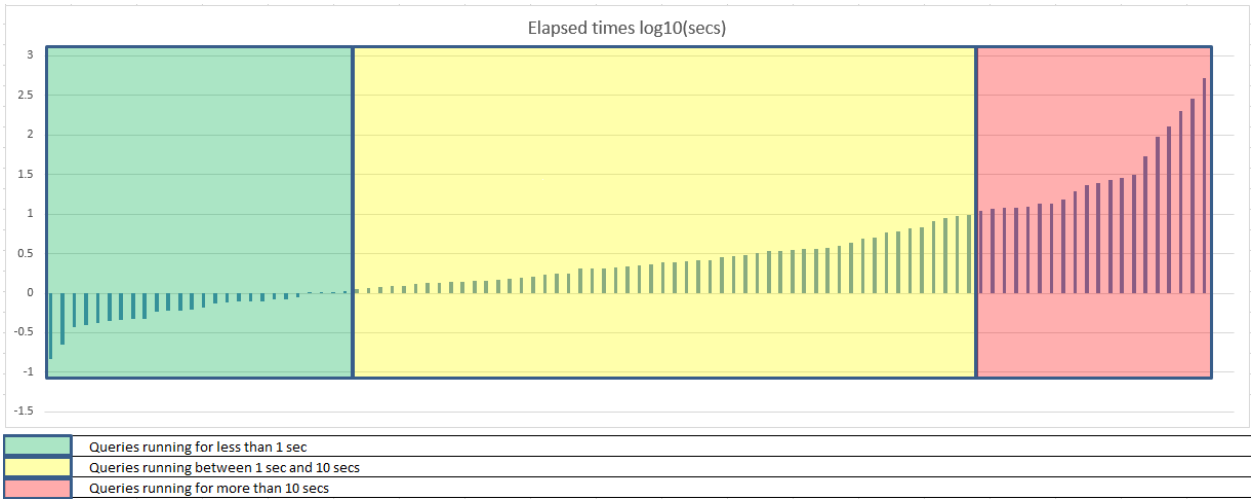

Figure 1. Logarithmic graph of runtimes of all 99 queries for 1 TB (each bar represents a unique query from the set of 99 queries).

Query optimization effort

We identified a few performance issues while executing some of the queries. A significant effort was put into evaluating alternate query plans and identifying optimal trace flags for queries that yield a better performance (Q2, Q14, Q38, Q59, Q78). These initial efforts led to a significant performance benefit of around 6x for the overall power run time. There is an ongoing effort to parallelize a handful of serial plans for a few of the single threaded queries that show potential for performance upside (Q9). The benchmark was thoroughly investigated for missing indices and adding these indices to the schema showed a performance benefit of around 25% to the power run time (Q23). Since this is a relatively new workload for single node execution, there are a few other use cases we are looking at that might give back even more performance.

Table 2 depicts the performance benefit on a 3 TB scale factor by using the above optimizations. INF for baseline highlights that some of the queries in the baseline run did not even finish unless specific trace flags were used (which offered a runtime of 7,544 seconds). The use of indexes further helped to improve the runtime when compared to the performance observed with trace flags.

	Total Power Run time (secs)
Baseline	INF
Traceflags	7,544
Indexes	6,654

Table 2. Impact of various optimizations on the elapsed time for power run phase (3 TB)

Data Scalability

Data scalability refers to the capability of the database and underlying system to offer consistent performance with respect to data size. The scalability in performance as we scale the database size from 1 TB to 3 TB is about 3.82x. Our initial results highlight how the scalability of the derivative workload under study is dominated by one of the long running queries (query67a) that is memory capacity sensitive.

Excluding this query, the scalability of the workload is about 2.71x for a 3x increase in scale factor which offers a clear direction for future optimizations. The bulk of the queries (~93%) in our experiments ran with complete parallelism with all 112 threads, this indicates the workload scales well with the increase in the number of cores.

Next, we highlight the usage pattern of the system database, TempDB. We observed it for both the 1 TB and 3 TB data sizes used in our study. We profiled the system while running with 1 TB dataset and configured the TempDB files to adequately handle the workload based on our studies with other data warehousing workloads. We recognized that since the workload is memory capacity sensitive, there would be a significant amount of total TempDB activity. *Total TempDB* which represents the sum of the data that is written (spilled) by all the queries to tempdb, is used as a measure of the stress on the I/O subsystem throughout the span of the workload. With the two data sets we observed the total tempdb scales almost linearly with the increase in data set size (3.3x). Furthermore, the maximum amount of data that is written to TempDB (Peak TempDB) by a query was also close to the total TempDB activity in our experiments. This further highlighted how only one of the queries (query67a) accounted for the majority of the TempDB activity. Table 3 and Figures 2,3 show the performance behavior (runtime, total tempdb, peak tempdb) for the two data sizes we used in our study.

Platform	Scale factor (GB)	Logical processor count	System Memory (TB)	SQL Max server memory (TB)	Runtime (secs)	Peak TempDB (GB)	Total TempDB (GB)
CLX-2S	1000	112	1.5	1.3	1740	971	1024
CLX-2S	3000	112	3	1.3	6660	2800	3380

Table 3. Performance Characteristics of TPC-DS derivative for 1 TB and 3 TB

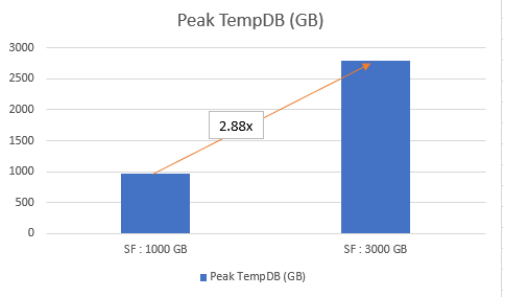

Figure 2. Runtime Scalability

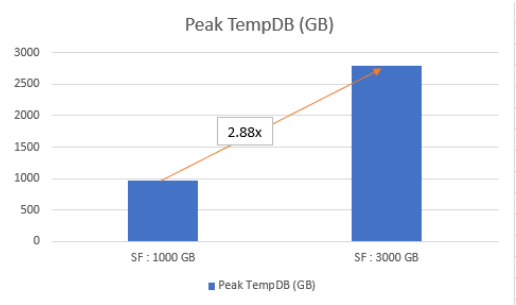

Figure 3. Peak TempDB Scalability

Query classification

We also characterized the queries based on resource utilization (CPU, memory, and I/O usage). I/O is further split into the footprint based on query execution (spill operations writing to TempDB in SQL Server) and the data requirement based I/O (reads from the database's data files).

- About a quarter of the queries in the workload depicted a CPU utilization of >60% in user space and there were only two queries that had a kernel utilization >5%.
- The queries that are sensitive to memory capacity are determined by the amount of spills a query incurs during execution. On 1 TB, about 15% of the queries spill into TempDB.
- Lastly, we label a query as I/O sensitive if it reads data from disk. In our workload these constitute around 60% of the queries.

Summary

In this paper, we presented our results of executing the complete set of 99 queries of a sophisticated analytical benchmark derived from TPC-DS, using advanced analytical capabilities of Microsoft® SQL Server 2019 and on a single-node configuration powered by the Intel® Xeon™ 8280L product family. While this does not constitute an official benchmark result by any means, the work in this paper does represent a novel industry case study. The study demonstrates that while these distributed, multi-node paradigms excel at large scale analytics but SQL-based engines on single-node configurations also offer a compelling story for large scale data sets.

Continuing our collaboration, Microsoft, and Intel plan to bring additional insights and innovation both on the software and hardware side driven by the optimization opportunities identified through benchmarks such as those derived from TPC-DS. Stay tuned for additional performance insights based on additional larger scale factors and a holistic view of database and system performance by including results from phases beyond power run.

References

[1] Nambiar, Raghunath Othayoth, and Meikel Poess. "The Making of TPC-DS." In *VLDB*, vol. 6, pp. 1049-1058. 2006.

[2] http://tpc.org/TPC_Documents_Current_Versions/download_programs/tools-download-request5.asp?bm_type=TPC-DS&bm_vers=2.13.0&mode=CURRENT-ONLY

[3] Poess, Meikel, Raghunath Othayoth Nambiar, and David Walrath. "Why You Should Run TPC-DS: A Workload Analysis." In *VLDB*, vol. 7, pp. 1138-1149. 2007.

[4] https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/microsoft-sql-database-analytics-paper.pdf

[5] https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-query-performance?view=sql-server-ver15

[6] http://tpc.org/tpcds/results/tpcds_results5.asp

[8] TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark: https://link.springer.com/chapter/10.1007/978-3-319-04936-6_5

[9] http://tpc.org/tpch/results/tpch_results5.asp

DISCLAIMERS

© 2021 Microsoft Corporation. All rights reserved.

This document is for informational purposes only and is provided “as-is”. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION IN THIS DOCUMENT.

Information and views expressed in this document, including URL and other Internet Web references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. In addition, some information may relate to pre-released product which may be substantially modified before it is commercially released.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document solely for your internal, reference purposes.