MemVerge

White Paper

SQL Databases and Memory Management

SQL Databases and Memory Management

Executive Summary

Performance is important for SQL databases because they are widely used to support critical business functions. Before SQL operations can run, data must be copied from disk to memory where the data is cached in pages. Under memory pressure, pages must be evicted from the cache to make room for new data. Every cache miss triggers a read from disk. Disk I/O is resource intensive and can cause performance degradation. Consequently, memory management is critical to maintaining SQL performance.

Increasing memory capacity will improve SQL performance when there is memory pressure but DRAM is expensive and limited in capacity. Intel's Optane DC Persistent Memory[1] is available in larger capacities at a lower cost than DRAM but has higher latency. Memory Machine, a software package from MemVerge,[2] can be used to manage a combination of DRAM and PMEM in such a way that the increased capacity can be used without incurring significant performance penalties.

This paper describes the results of benchmark tests run against Microsoft's SQL Server and the open source MySQL. Results show that, by using Memory Machine, customers will have options to reduce cost by configuring less DRAM or to increase performance by starting additional SQL instances using the same amount of DRAM.

Introduction

In a Relational Database (RDB), data is organized in tables of columns (attributes) and rows (records) with a unique key identifying each row. The concept was originated by Edgar Codd, an IBM mathematician, in 1970.[3] Databases existed before then but they required specialized programming skills in order to access the data. The IBM team also developed Structured Query Language (SQL) as a tool for querying and managing RDBs. SQL has become ubiquitous so that today RDBs are commonly referred to as SQL databases.

The most commonly used database systems[4] are all SQL based although other types are growing in popularity (for example, MongoDB and Redis are in the top ten). The three most popular database systems are:

1. Oracle
2. MySQL
3. Microsoft SQL Server

SQL databases can be large (single instances of tens of TBs have been reported) and in general are larger than typical DRAM capacity. Database transactions (queries for example) will require blocks of data to be cached in 8 KB memory pages. Movement of data between memory and SSDs (or rotational disks) involves the CPU in I/O operations for which there is a significant latency penalty. The key to SQL performance is the management of the memory buffer pool.

Scale of Computer Latencies

Latency increases exponentially as the target media move further from the memory bus To illustrate this, access times for random read requests are displayed in Table 1.[5] [6] Specific numbers change as new products are released but the overall scale remains the same.

Table 1. Scale of Computer Latencies

Target	Latency for random read access (nanoseconds)	Normalized scale (DRAM access time = 1)
DRAM	100	1
PMEM (no DRAM cache)	300	3
NVMe SSD (3D Xpoint)	10 000	100
SATA SSD (NAND)	100 000	1 000

SQL Performance Optimization

SQL databases have become highly optimized as a result of their wide use in nearly all industries, government agencies, and academic institutions. Tools exist for tuning SQL queries to determine the most efficient execution path. Table indexes are tailored for particular search types; for example, columnstore indexes for large table scans can improve performance by 10x over traditional row-oriented storage.[7] Limited DRAM capacity remains an insurmountable performance barrier.

Memory Machine and Persistent Memory

Intel's Optane DC Persistent Memory is available in 128GB, 256GB and 512GB capacities and can be installed alongside traditional DDR4 DIMMs. In a 2-socket server, it is theoretically possible to install up to 9 TB of PMEM. Although PMEM has much larger capacity than DRAM, PMEM has natively higher latency (see Table 1). The challenge is to take advantage of the extra capacity and persistence of PMEM while ameliorating the higher latency.

Memory Machine is able to achieve DRAM-like performance, on average, by using DRAM and PMEM in a virtualized, two-tier memory hierarchy. Applications can run without modification while in the background Memory Machine uses a sophisticated memory management algorithm to place data dynamically in DRAM or PMEM.

Memory Machine and SQL Database Engines

If the entire database (or all the data that needs to be accessed) can be cached in DRAM, there will be no performance improvement by adding PMEM. When there is memory pressure, performance can be improved by adding memory. When Memory Machine is installed, PMEM can be added instead of more expensive DRAM. By using Memory Machine with PMEM, less DRAM capacity is consumed per SQL instance, thereby freeing up DRAM resources for other processes or allowing additional SQL instances to be started.

Test Results: Microsoft SQL Server

SQL Server is Microsoft's flagship RDB product and can be run on–premises or in the cloud as Azure SQL Server. Benchmark tests were run on a single server using the following set-up.

Server platform
CPU: 2 x Intel Xeon Gold 5220 @ 2.20GHz (18 cores per socket)
DRAM: 12 x 16GB = 192 GB DDR4
PMEM: 8 x 128GB = 1024 GB Intel Optane DC Persistent Memory

Software
Linux: CentOS 8.1.1911
RDB: Microsoft SQL Server2019
Benchmark: TPC-C with HammerDB

HammerDB[8] is an open source benchmarking and load-testing tool used to measure performance of SQL databases. TPC[9] is an industry body that defines benchmarks and its specifications are recognized by all of the leading database vendors. TPC-C is the benchmark published by the TPC for Online Transaction Processing (OLTP) and HammerDB includes a workload derived from TPC-C. The workload includes multiple reads and writes from a group of concurrent users applied to a collection of data warehouses.

Output from HammerDB TPC-C tests using 800 data warehouses is shown in Figure 1. In this figure, Total Memory Usage (of 10 GB, 20 GB, etc.) refers to the sum of DRAM+PMEM configured for the particular test run. For example, DRAM:PMEM ratio of 1:4 in the 10 GB case means that SQL Server had access to 10GB of memory of which 2 GB was DRAM and 8 GB was PMEM.

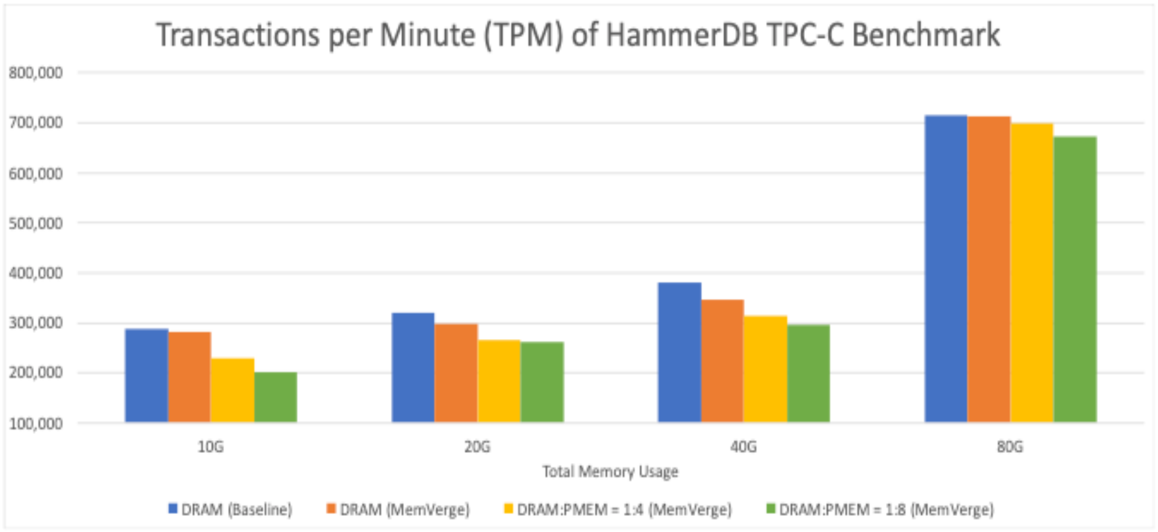

Figure 1. HammerDB TPC-C benchmark applied to SQL Server 2019

The results show that with insufficient memory (less than 40 GB), transactions per minute (TPM) never reach 400k even when using DRAM only. When 80 GB DRAM is used, TPM slightly exceeds 700k and shows that Memory Machine imposes negligible overhead. When DRAM is reduced to 16 GB (80% reduction) and the balance made up of PMEM, Memory Machine is able to maintain 700k TPM. With a 90% reduction in DRAM, TPM is reduced by less than 5%.

Reducing DRAM by these amounts also reduces the associated cost. A cost model showed that using 80GB of memory in a DRAM:PMEM ratio of 1:8 reduced the memory cost by about 50%. Prices are subject to change so this model is for illustrative purposes only.

If maximizing performance is the goal, then these results suggest that the number of SQL Server instances could be increased by 5x to 9x by using Memory Machine with PMEM in an 80 GB memory footprint. SQL Server is CPU-intensive so care must be taken when scaling up significantly because other system factors may become the bottleneck.

Test Results: MySQL

MySQL is an open source SQL database (commercial versions are available from Oracle). For these tests, MySQL was installed in a KVM-based VM (configured with 8vCPUs and 16GB memory) and subjected to the Sysbench[10] benchmark. The VM was unaware of the distinction between DRAM and PMEM because the DRAM was allocated dynamically by Memory Machine (in addition to managing a large pool of PMEM). Results are shown in Figure 2.

Two database sizes were used: small database allowed the entire database to be cached in DRAM, and large database did not. Baseline case was 16 GB DRAM only. For the small database, the results were within 8% of the baseline case when DRAM was reduced to 4 GB or 2 GB. For the large database (i.e., when the DRAM was under memory pressure), the results were 5% better with Memory Machine reserving 16 GB DRAM for the VM in addition to managing the PMEM. Reserving 4GB (or

2GB) of DRAM when querying the larger database caused the performance to degrade by only 3% (or 6%).

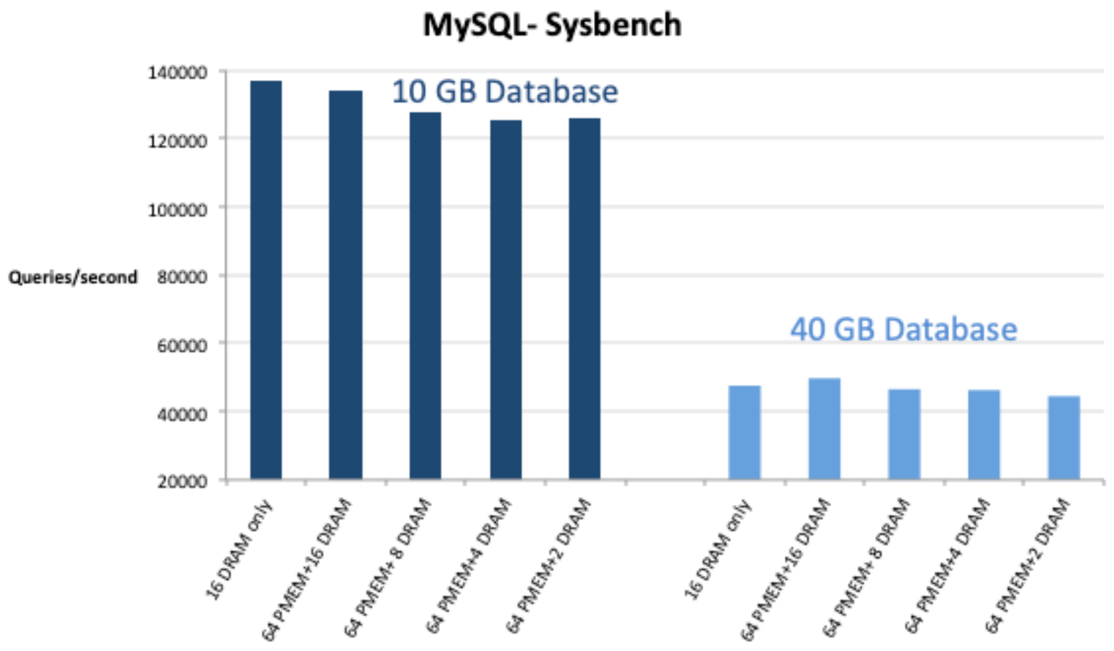

Figure 2. MySQL performance measured by Sysbench

As in the SQL Server case, the customer has the option to reduce cost by configuring less DRAM capacity or to increase performance by starting additional MySQL instances using the same amount of DRAM.

Test Results: Memory Management

SQL database engines are configured with upper and lower limits to the memory allocated. Not all the memory has to be allocated and the actual amount in use is managed dynamically. A memory management process will release unused memory back to the OS and claim memory should it be needed. Operations such as join, merge or sort applied to large tables will require large amounts of memory. If insufficient memory is available, a memory timeout error will be generated.

To demonstrate this behavior, MySQL was started in a VM with access to 32 GB DRAM and 128 GB PMEM. Sysbench was used to measure how performance changed as the memory buffer size changed. The results are shown in Figure 3.

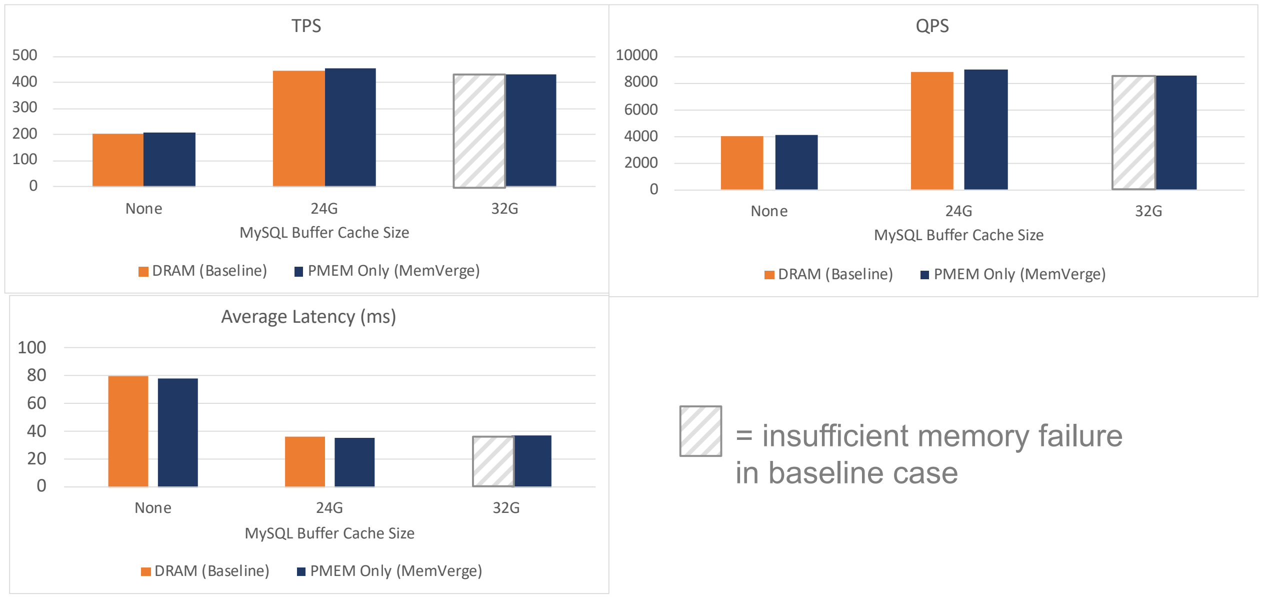

Figure 3. MySQL performance with different memory buffer sizes

When no memory buffer is allowed, all SQL queries require disk I/O. Average latency is more than doubled. With the buffer comprising PMEM only (no DRAM), Memory Machine is able to achieve the same performance as the DRAM only case. When the buffer size was set to 32 GB DRAM only, the MySQL instance crashed because there was insufficient memory to support MySQL in addition to the OS and other background processes.

Conclusion

SQL optimization continues because SQL databases are widely used to support critical business functions. Disk I/O is resource intensive and can cause performance degradation under memory pressure, highlighting the importance of memory management. By allowing Memory Machine to manage a combined pool of DRAM and PMEM, memory pressure can be reduced. Cost can be reduced by reducing the DRAM capacity allocated or performance can be increased by running multiple SQL instances with the same amount of DRAM.

[1] https://www.intel.com/content/www/us/en/products/docs/memory-storage/optane-persistent-memory/optane-persistent-memory-200-series-brief.html
[2] https://memverge.com
[3] https://www.ibm.com/ibm/history/ibm100/us/en/icons/reldb/
[4] https://db-engines.com/en/ranking
[5] https://arxiv.org/pdf/1903.05714.pdf
[6] https://www.micron.com/products/advanced-solutions/3d-xpoint-technology/x100
[7] https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview?view=sql-server-ver15
[8] https://www.hammerdb.com
[9] http://www.tpc.org
[10] https://github.com/akopytov/sysbench

Learn More

Big Memory

IDC Big Memory Definition and PMEM Forecast Presentation

IDC Big Memory Definition and PMEM Forecast Video

The Next Platform: The Era of Big Memory is Upon Us

Webinar: Breakthroughs in Big Memory

Intel Podcast: Big Memory Software Defined Controller

Memory Machine Software

The Skinny on Memory Machine

1-Page Memory Machine Data Sheet

Demo: Creating Clones of Redis VMs in Microsoft Azure

Demo: Memory Machine Software Capabilities: Memory Snapshots and Managing from GUI and Command Line

Demo: Cloning an 800GB kdb+ Database in Seconds

Demo: See kdb+ in-memory on AWS run faster with Memory Machine software

Intel Optane Persistent Memory

Intel Optane Persistent Memory

MemVerge

MemVerge Corporate Brochure

MemVerge

@memverge

@memverge