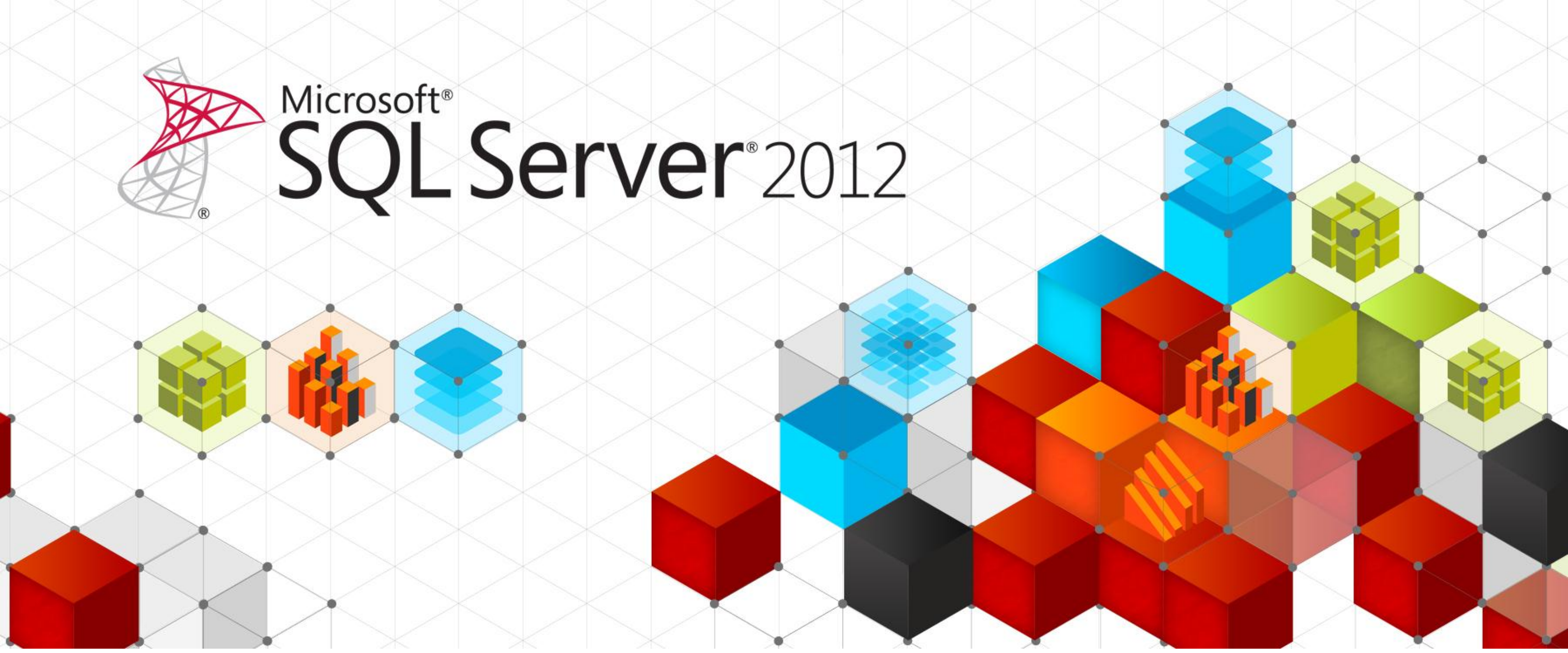

xVelocity in SQL Server 2012 Delivers Order-of-Magnitude Gains for Data Warehouse Queries

Highlights

- **xVelocity delivers break-through gains for data warehouse queries**
- **First memory optimized columnar data store integrated into a mainstream relational database engine**
- **Benchmark tests demonstrate order-of-magnitude speedups**

Introduction

The new xVelocity memory optimized column store index feature in SQL Server 2012 delivers breakthrough performance gains for data warehouse queries. SQL Server 2012 is the first mainstream RDBMS to integrate a memory optimized columnar data storage capability into the relational database engine. Using a combination of compression, algorithms enhanced for modern CPU/memory architecture, and highly parallel intra-query execution, xVelocity can speed up query execution times by 4x, 10x, and even 100x, simply by adding column store indexes to the fact tables of a star-schema data warehouse*.

Benchmark Results

To gauge the benefits of xVelocity, Microsoft engineers measured a collection of complex data warehousing queries running with and without column store indexes on a large data warehouse. This datasheet highlights key findings of the benchmark results and various usage considerations.

From test results, queries that benefit from the memory optimized column store index showed dramatic gains. Queries that ran for minutes or tens of minutes completed in seconds with the new index. However, not all queries showed these benefits depending on their characteristics.

The graph Figure 2 on the next page shows the speedup in elapsed times for nine queries. It examines performance with a cold buffer pool (no pages in memory) and also a warm buffer pool (after previous queries had run). A cold buffer pool requires that all data be read into memory, whereas a warm buffer pool is a purer indicator of scalability and CPU efficiency.

With the cold buffer pool, speedups ranged from 0.9 (a small slowdown) to

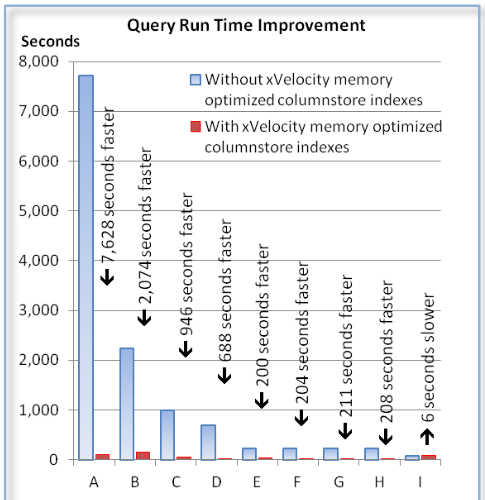

Figure 1: The graph compares elapsed time of nine complex DW queries measured before and after adding column store indexes to the fact tables.

107x, with most queries falling in the **8-20x** range. For the warm buffer pool, speedups were more disparate with two queries seeing no speedup, three queries in the **4-10x** range, and four queries showing speedups in the range from **257 to 615x**.

Data Warehouse Stats

The characteristics of the data warehouse used in this benchmarking exercise consisted of:

- 1 terabyte of base table data
- 24 tables total
  - 7 fact tables
  - 17 dimension tables
- 6+ billion rows
- Star schema

Hardware Configuration

The server used for the tests was a high-end industry standard server, consisting of:

- CPUs
  - Intel Xeon E7 4870 2.4GHz
  - 4 proc, 40 cores, 80 threads
- Memory: 256 GB
- Data Disks
  - 2 RAID5 stripes each with
  - 15 x 136GB SAS

Usage Considerations

While xVelocity memory optimized column store indexes can provide big gains, there are some usage aspects to consider.

One of the ways that Column store indexes are memory optimized is through efficient use of the buffer pool. Column store indexes are paged in and out of the buffer pool similarly to row store pages, unlike other columnar implementations that only work if the entire dataset fits in memory. Since column store indexes are compressed and grouped by column, they typically have a much smaller memory footprint than the row store data.

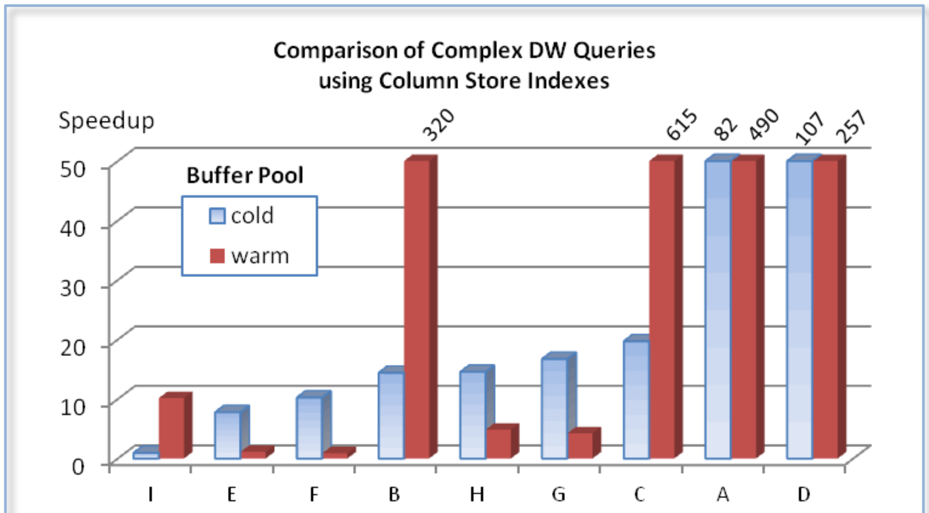

Figure 2: The graph shows the speedup in elapsed time of nine complex DW queries measured after adding column store indexes to the fact tables. The queries were run with the buffer pool empty ("cold") and after warm-up runs ("warm"). Speedups greater than 50x are labeled above the columns.

Column store indexes are read-only; updates aren't allowed for the indexes or base tables. To update the base table, drop or disable the column store index, do updates, and then rebuild the index.

Column store indexes are most effective with data warehouses using a star schema. You can get breakthrough performance by designing and optimizing for column store indexes. But even without schema or application changes, column store indexes can improve performance which then helps provide higher ROI on new and existing systems.

A common technique to optimize DW performance is to build indexes and aggregates that are tailored to the particular needs of the users. This carries two distinct disadvantages: (1) choosing the indexes and aggregates requires both insight and expertise, and (2) aggregates aren't general purpose. In contrast, column store indexes retain all of the fine-grain detail of the data, allowing users to do arbitrary ad hoc queries with consistently fast performance.

Learn More

To learn more, enter "columnstore" in the search box at msdn.microsoft.com.

Join the conversation

www.microsoft.com/sqlserver

Or follow us! /sqlserver

[*] The performance of any particular application depends on many factors. Results may vary.