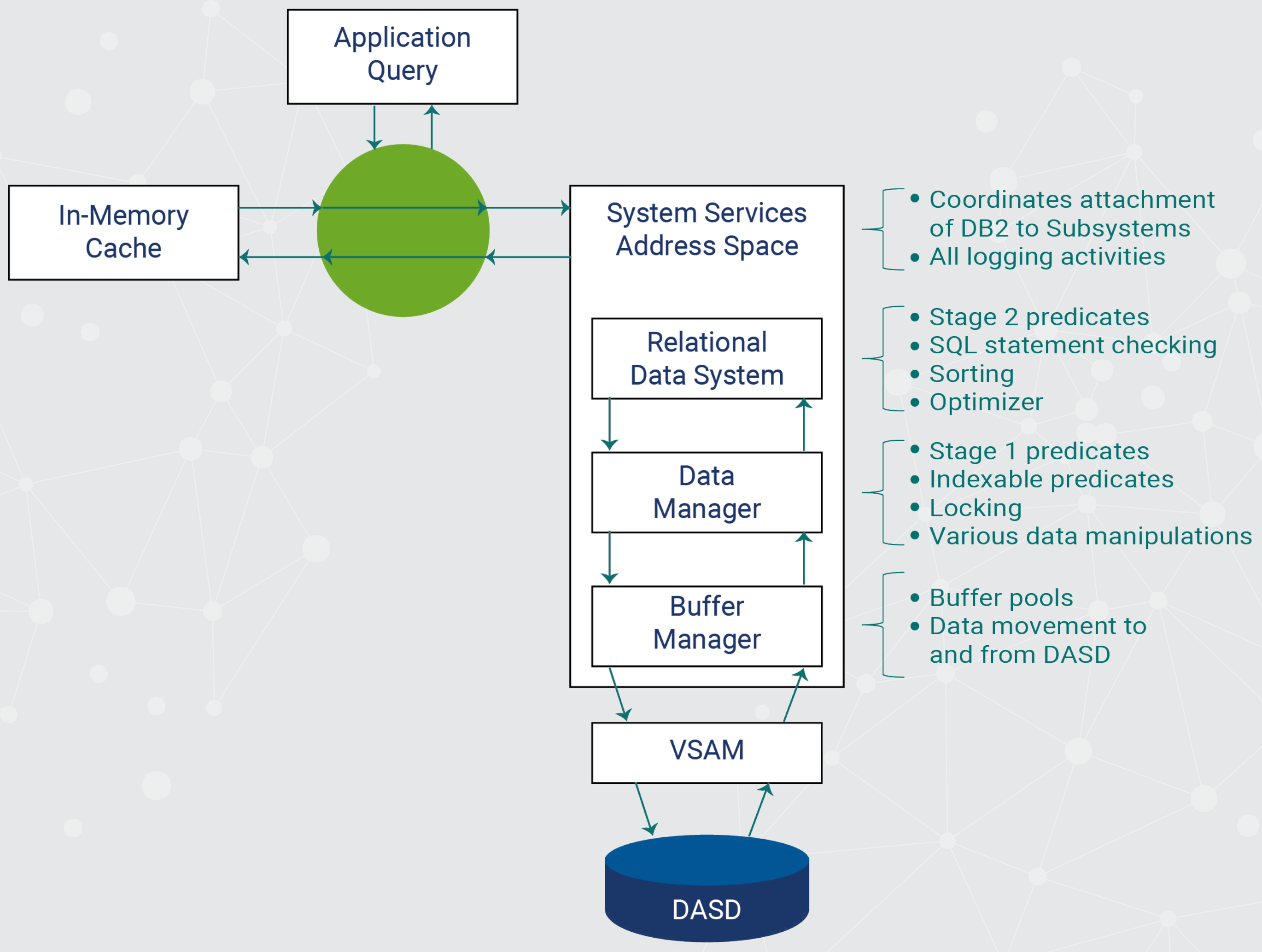
Unleashing Db2 SQL Workload Performance Using In-Memory Techniques: Exploring SQL Query Result Set Caching

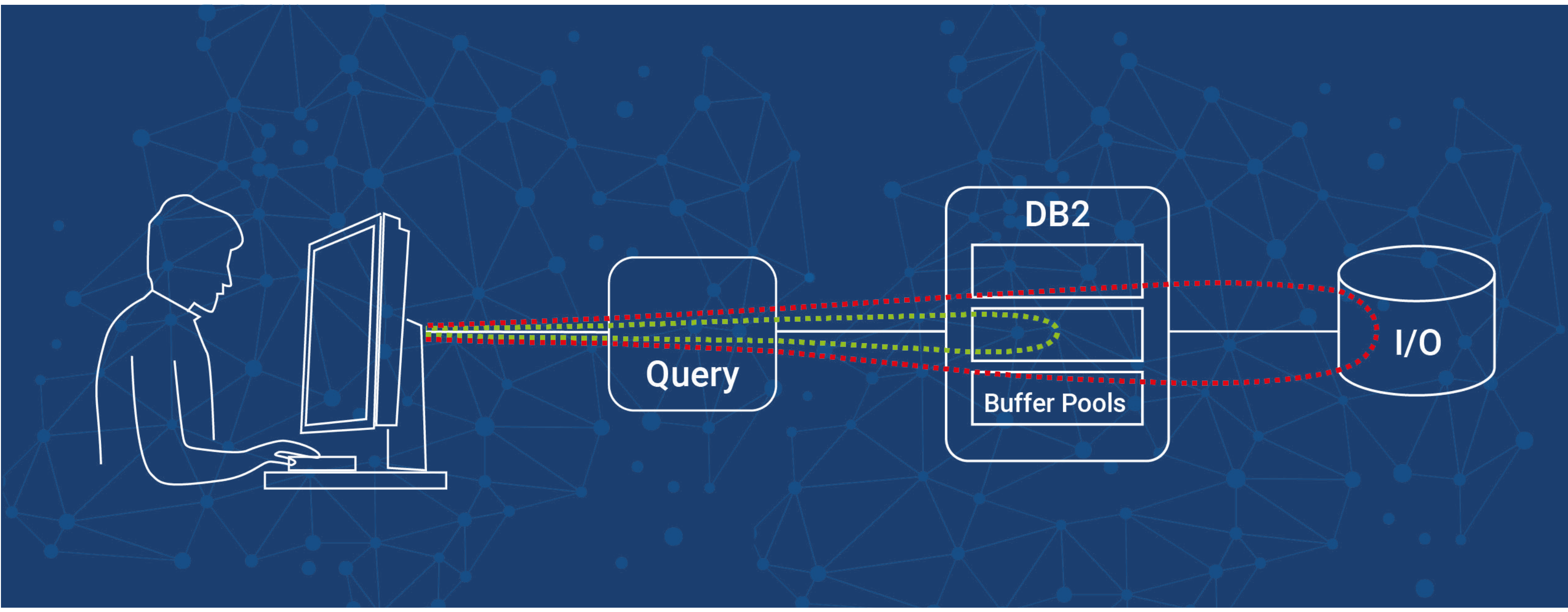
By Craig S. Mullins

In the realm of database management, performance optimization is a perpetual pursuit. As data volumes surge and query complexities escalate, innovative techniques are required to ensure optimal performance. The use of memory to cache data is a long-standing technique used by database management systems in the form of buffer pools. Another emerging technique for improving query performance is SQL query result set caching.

As the diagram below illustrates, Db2 SQL query execution traverses a long code path and consumes a good deal of CPU. Using memory to improve query performance is a crucial strategy for improving database performance and reducing resource consumption. In this article, we delve into in-memory data caching techniques, highlighting their benefits, differences, and real-world impact on system efficiency.



Db2 Buffer Pool Processing:
Leveraging Memory for Performance Gains



Db2 Buffer pools are dedicated areas of memory that are used to store data pages or blocks that are frequently accessed by queries. All requested data is accessed via the buffer pools. The first time that data is requested, it is retrieved from disk and placed in the buffer pool before being passed along to the requesting user or application. Every time a query requires data, the DBMS checks if the required data pages are already in the buffer pool. If so, it retrieves the data from those pages, instead of requesting an I/O. This results in a generally favorable overall impact on performance.

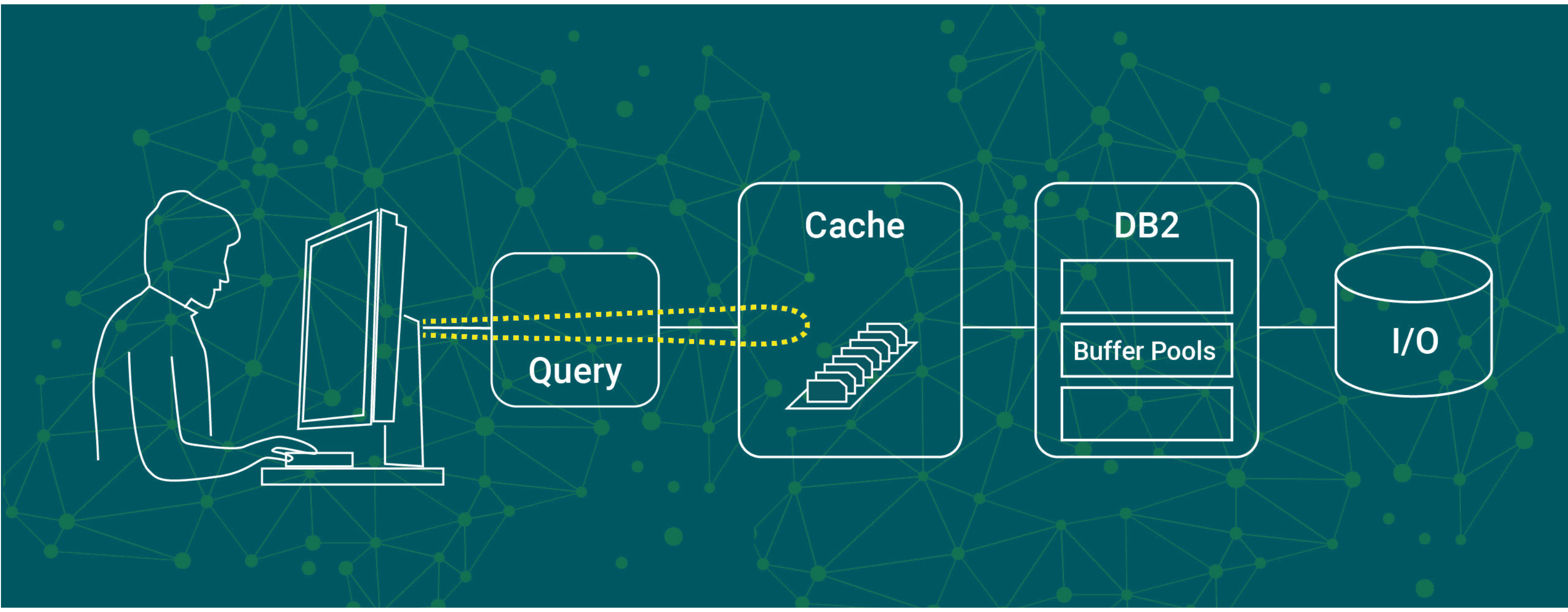
Db2 buffer pool processing focuses on optimizing data access by caching frequently accessed data pages in memory buffers. When a query requires data that is already in the buffer pool, the database retrieves it from memory instead of fetching it from disk.

Key benefits of buffer pool processing include:

- **Faster Data Access:** By keeping frequently accessed data in memory, buffer pool processing minimizes disk I/O operations, resulting in faster data retrieval and query execution times.
- **Reduced Latency:** Memory-resident data reduces latency compared to disk-based access, leading to improved application responsiveness and user satisfaction.
- **Resource Optimization:** Buffer pool processing optimizes resource utilization by prioritizing memory for frequently accessed data, thereby enhancing overall system performance and efficiency.
- **Concurrency Control:** Buffer pools facilitate efficient concurrency control mechanisms, ensuring data consistency and integrity in multi-user database environments.

It is for these reasons that buffer pool processing is a crucial feature of the Db2 DBMS.

SQL Query Result Set Caching: Enhancing Speed and Efficiency



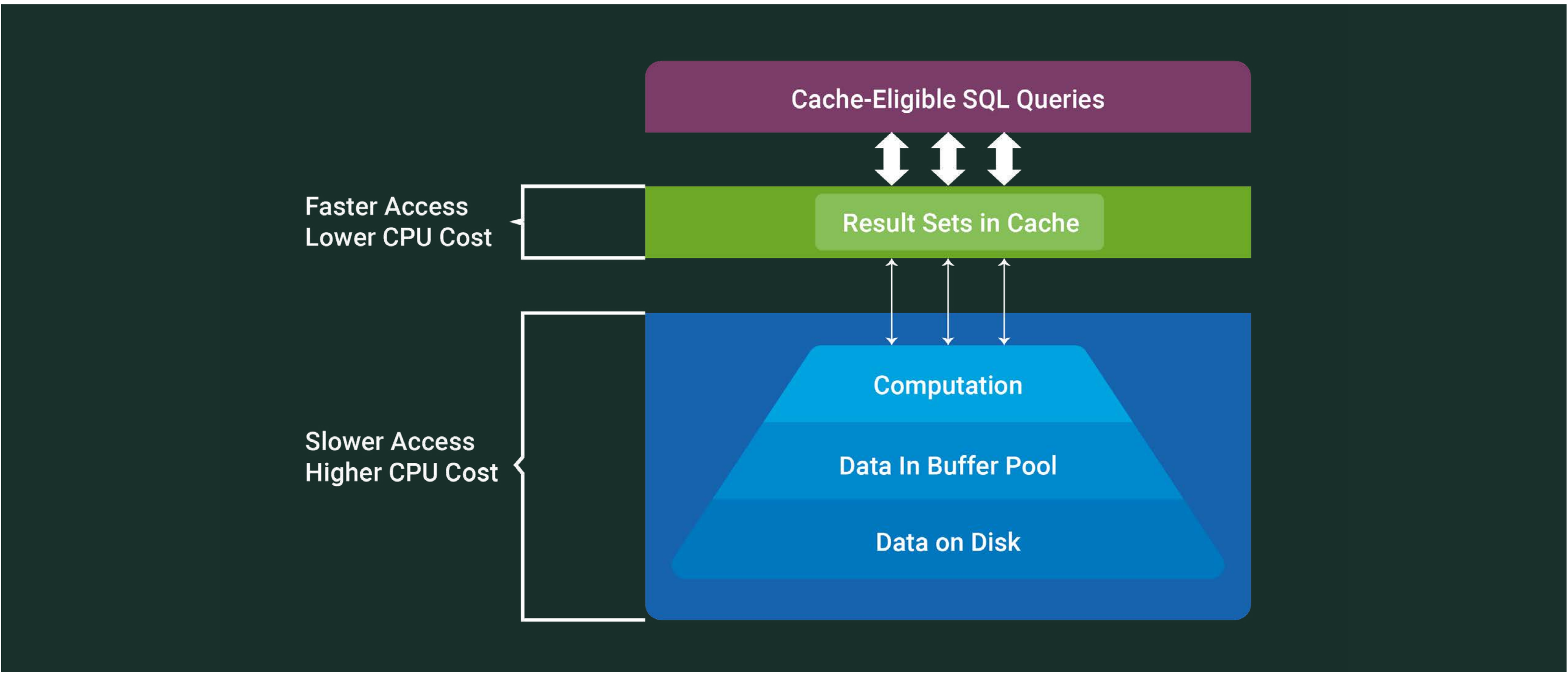
Another approach to caching data in memory is to cache the results of SQL queries. This is known as “SQL query result set caching” and it involves storing the results of frequently issued queries in memory. For targeted queries, the first time the query runs, the SQL query result set is saved to an in-memory cache. When an identical query is subsequently issued, the system retrieves the result set from the cache, pre-empting re-execution of the query by the database.

This approach offers several advantages:

- **Reduced Database Load:** By retrieving data from cache rather than re-executing queries, SQL query result set caching reduces the workload on the database, minimizing resource utilization and improving overall system performance.
- **Faster Response Times:** Cached result sets lead to faster responses to application queries, enhancing application performance and user experience. This is particularly beneficial for read-heavy workloads where data retrieval speed is paramount.
- **Improved Scalability:** Caching transforms a portion of the normal database workload to simple result set retrieval. This allows databases to handle higher query volumes without experiencing performance degradation, making it an effective strategy for scaling systems as workload volumes grow.
- **Cost Efficiency:** Reduced database load and improved performance translate to cost savings. Organizations can achieve optimal performance without incurring additional hardware and software costs.

SQL query result set caching operates at a higher abstraction level than buffer pool data page caching. This approach caches SQL query result sets exactly as Db2 would deliver them to the application. For SQL queries whose result sets reside in cache, the SQL statement need not be executed at all.

Contrasting Approaches:
SQL Query Result Set Caching vs. Buffer Pool Processing



While both SQL query result set caching and buffer pool processing aim to improve database performance, they operate at different levels within the data access hierarchy. SQL query result set caching focuses on caching query results, reducing database load, and enhancing query response times. On the other hand, buffer pool processing optimizes data access by caching data pages in memory, minimizing disk I/O and latency.

Consider some of the advantages of caching SQL query results versus caching data pages in buffer pools:

- **Bypass SQL Operations:** By caching the exact data required to satisfy the query, an SQL result set caching approach need not execute a repetitive query again until or unless the underlying data changes. By avoiding Db2 execution of the SQL altogether, a lot of overhead is saved. With buffer pools this is never an option as pages of data are cached, not actual SQL results. With buffer pools, repetitive queries are executed repetitively.
- **Buffer Pools are Volatile:** The capacity of the buffer pools will never approach the capacity of data on disk. Data pages are moved in and out of buffers all the time based on on-going activity. When data is required, the page must be moved into the buffer pool. It is quite possible that a page cached in the buffer pool is used only once before it is moved out to make room for other data. This is not the case with result set caching where you have the option to cache only frequently requested result sets, and where the storage required to cache frequently requested result sets is much less than the space required to cache all requested data pages.

- **Minimize Db2 data and I/O processing:** A lot of processing is required to satisfy an SQL query and return results to the requester. As illustrated in the diagram on the first page, this includes I/O processing (itself an expensive operation), GETPAGE processing, processing and managing data throughout the internals of the DBMS (such as the Buffer Manager, Data Manager, and Relational Data System for Db2 for z/OS), and more. For a result set returned from cache, SQL query result set caching eliminates all this processing.
- **No Locking Required:** Sometimes database locking can be a significant impediment to database performance and application processing. When applications use cached query result sets, no locking is required at all for those queries as the DBMS does not execute them. The data is simply returned from cache.

In practical terms, SQL query result set caching is beneficial for read-heavy workloads with recurring queries, where caching entire result sets can significantly boost performance. Buffer pool processing, on the other hand, is essential for optimizing data access at the page level, ensuring efficient use of memory resources and minimizing disk-related bottlenecks.

In the Db2 world, an optimal approach can be implemented by combining an SQL query result cache with standard Db2 buffer pool processing. Combining buffer pool processing and SQL query result set caching can work to use the memory at your disposal in the most optimal manner for each type of database request.

Use Cases for SQL Query Result Set Caching:

SQL query result set caching is beneficial for scenarios where the same queries are executed frequently, and where the underlying data changes infrequently, allowing for efficient reuse of cached results. For example, batch programs and online transaction workloads having repetitive queries that access lookup, code, reference, and other relatively stable data types will see significantly enhanced performance.

Consider a query that runs many millions of times a day repeating frequently across a range of host variables. Now imagine all that CPU intensive Db2 processing and I/O wiped away by storing and accessing the result sets from cache.

Real-World Impact and Best Practices

The adoption of SQL query result set caching has delivered tangible performance improvements for organizations across industries.

Best practices include:

- **Balancing Caching Strategies:** Strike a balance between SQL query result set caching and buffer pool processing based on workload characteristics, data access patterns, and performance objectives.
- **Identifying Performance Hotspots:** Analyze query patterns and data access patterns to identify queries that would benefit most from result set caching.
- **Optimizing Cache Management:** Implement effective cache management policies to ensure optimal use of memory resources and cache strategies.
- **Monitoring and Fine-Tuning:** Although SQL query result set caching should not require in-depth monitoring; it is wise to periodically monitor your applications for cache utilization and determine if all of the queries being cached are still optimal and/or viable. For example, did a business process change result in too-frequent update of the table data underlying cached result sets? Is the query no longer running as frequently as it did in the past? Did application changes introduce new queries that will benefit from result set caching?

You should look for a Db2 SQL query result set caching solution that offers a way to survey your existing environment for queries that will benefit from caching. Furthermore, you should look for a solution that is non-invasive. You do not want to have to change your database environment, database structures, application code, or JCL to implement an SQL query result set cache.

An effective solution should self-manage and dynamically build the result set cache without the need to define and maintain data synchronization processes.

Query result set caching solutions should come with a built-in mechanism for identifying underlying changes. When a change is about to be made to a table underlying cached result sets, those cached result sets should immediately be marked as invalidated and dependent queries passed to Db2 for regular processing. Be sure that there are no back doors where data changes could go undetected. For example, the tool should be sensitive to the execution of database utilities – such as Load and Recover - that change data in tables.

One such tool that implements SQL result set caching for Db2 for z/OS is QuickSelect for Db2 from Log-On Software.

Summary

In conclusion, SQL query result set caching and buffer pool processing are powerful techniques for enhancing database performance, reducing latency, and optimizing resource utilization. By understanding the nuances of these approaches and implementing best practices, organizations can unlock the full potential of their databases, delivering superior performance, scalability, and user experience.

About Craig Mullins

Craig S. Mullins is the president and principal consultant of Mullins Consulting, Inc., an independent consulting and strategy firm specializing in database management and mainframe systems. Craig has been involved with Db2 for z/OS since Version 1 and has extensive experience as an application developer, DBA, and instructor. He has been recognized as an IBM Gold Consultant and IBM Champion for Data and AI by IBM, and as an Influential Mainframer by Planet Mainframe. Craig is the author of several books, including the industry-leading "DB2 Developer's Guide" on Db2 for z/OS and "Database Administration: The Complete Guide to DBA Practices and Procedures," the only book on heterogeneous database administration. Craig can be contacted at http://www.MullinsConsulting.com

Additional information on QuickSelect for Db2 can be found at https://log-on.com/quickselect-for-db2-performance/ or by emailing to ask@log-on.com