Pipelining Vector-Based Statistical Functions with eXtremeDB®

**Pipelining for Low Latency**

Pipelining is the programming technique in eXtremeDB for HPC that accelerates processing by combining the database system’s vector-based statistical functions into assembly lines of processing for market data, with the output of one function becoming input for the next. Calculations are pipelined in order to keep data within CPU cache during its transformation by multiple functions. Without pipelining, interim results from each function would be transferred back and forth between CPU cache and main memory, imposing significant latency due to the relatively lower-bandwidth front side bus (FSB) or quick path interconnect (QPI) between the two.

eXtremeDB for HPC offers pipelining in industry-standard SQL or in any of its other supported languages (Python, C/C++, Java, C#).

**Columnar vs. Row-Based Data Handling**

Pipelining builds on eXtremeDB for HPC’s support for columnar data handling. Database management systems (DBMSs) typically store data in tables consisting of rows and columns, with each row containing one instance of each column value. A conventional DBMS accumulates rows into database pages for processing, as shown in Figure 1 below. When an application needs to process a single column of data (whose elements would be in multiple rows) a different layout is more efficient. This columnar layout accelerates time series analysis (including market data analysis) by storing (and subsequently fetching) data in a column-by-column fashion on a database page, as shown in Figure 2.

![Figure 1](image.png)

Figure 1. Relational databases typically store data in tables consisting of rows and columns, and transfer data row-by-row between storage, main memory and CPU cache.

**Pipelining for High Performance**

Here’s how pipelining works. Let’s say the application needs to calculate 5-day and 21-day moving averages for a stock and detect the points where the faster moving average (5-day) crosses over or under the slower one (21-day).

This is accomplished below in SQL, using eXtremeDB for HPC’s vector-based statistical functions as expressions in a SELECT statement:

```sql
SELECT seq_map(Close,
    seq_cross(seq_sub(
        seq_window_agg_avg(Close, 5),
        seq_window_agg_avg(Close, 21)), 1))
FROM Security
WHERE symbol = 'IBM';
```

1. Two invocations of `seq_window_agg_avg` execute over the closing price sequence to obtain 5-day and 21-day moving averages.
2. The function `seq_sub` subtracts 21- from 5-day moving averages;
3. The result “feeds” a fourth function, `seq_cross`, to identify where the 5- and 21-day moving averages cross.
4. Finally, the function `seq_map` maps the crossovers to the original ‘Close’ sequence, returning closing prices where the moving averages crossed.

Columnar handling results in faster performance because only the column(s) of interest (closing prices, in our example) are brought into CPU cache at the start of the operation. In contrast, conventional row-wise handling would bring database pages consisting of entire rows with all their columns into CPU cache.
Even more significantly, this approach eliminates the need to create, populate and query temporary tables outside CPU cache (i.e., in main memory), as would be required by other database systems and vector-based programming languages to manage interim results of processing (for example, no table is needed to contain 5-day moving averages, 21-day moving averages, etc.).

The next step populates the temp table ‘sub’ with the result of subtracting the two moving averages:

```
INSERT INTO sub SELECT 5day - 21day FROM mavg;
```

From here, “normal” SQL can go no further. Code in a language such as C/C++ is needed to get crossover positions.

Clearly, using the traditional SQL DBMS requires a lot more code. But more importantly, all temporary (transient) results used in processing have to be created as output that is external to CPU cache, in temporary tables. This results in multiple trips “across the transom” to fetch data into CPU cache, and then to write it back into the temporary table. This occurs for each temporary table.

Without pipelining, the overhead due to traffic across the QPI or FSB, which is several times slower than CPU cache, is enormous. The delay generated by these transfers is multiplied by the number of pages required to move a given set of data back and forth. In other words, processing by the SQL DBMS isn’t just 3-4 times slower than the approach using pipelining. It is 3-4 times slower times the number of pages that have to cross the QPI/FSB. In contrast, pipelining reduces the number of transfers for interim results to zero.

### DBMS Optimized for Capital Markets

eXtremeDB for HPC features that help break through the financial IT database bottleneck include:

- A GUI database monitor for tuning and optimization
- Flexible storage, with a core in-memory database system (IMDS) architecture as well as optional and selective persistent storage
- Powerful distributed computing options including Distributed Query Processing, Cluster and High Availability capabilities
- Transactional – supports ACID properties
- Language support includes standard SQL/ODBC/JDBC, Python, C/C++, Java and C# (.NET)
- Event notifications “inform” the application when something of interest in the database changes
- Multi-version concurrency control (MVCC) transaction manager accelerates multi-threaded applications on multi-core hardware
- Security: page level cyclic redundancy check (CRC) detects unauthorized changes and RC4 encryption blocks tampering
- Optional run-length encoding (RLE) compression to cut application latency and storage requirements

Learn more at www.mcobject.com