Pipelining Vector-Based Statistical Functions with *eXtreme*DB®

With pipelining, the risk management proof-of-concept achieved 950 million calculations/second on a 4 host server, pushing the boundary of current risk technology.

-- McObject, Fultech and Kove financial risk solution

*eXtreme*DB for HPC, the low-latency database management system for real-time capital markets systems.

Pipelining for Low Latency

Pipelining is the programming technique in *eXtreme*DB Financial Edition 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.

*eXtreme*DB Financial Edition 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 *eXtreme*DB Financial Edition'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.

Open	Close	Vol	Date
204	205	200000	20120410
202	203	150000	20120411
204	205	300000	20120412
...	...	...	...
203	204	250000	20130624
202	203	400000	20130625

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.

*eXtreme*DB Financial Edition implements columnar handling via its 'sequence' data type – examples of sequences include the Open, Close, Volume and Date columns in Figure 2. Sequences exist alongside non-columnar data types in database records, enabling the most efficient data design.

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 *eXtreme*DB Financial Edition's vector-based statistical functions as expressions in a SELECT statement:

```
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.

McObject LLC
33309 1st Way South
Suite A-208
Federal Way, WA
98003

Phone:+1 425 888 8505
info@mcobject.com
www.mcobject.com

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.).

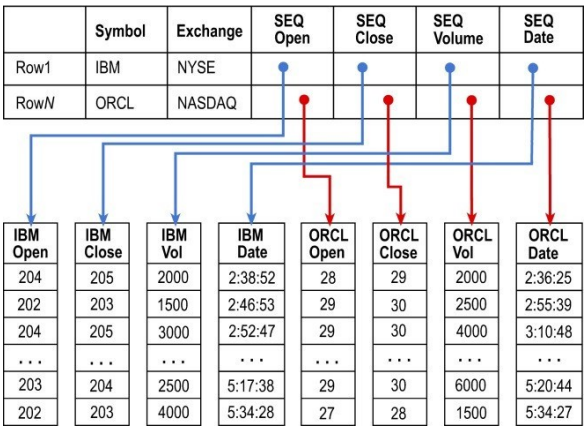

Figure 2. With a columnar approach, tables' columns are the basic unit for constructing the database pages used for DBMS input/output.

McObject's technology achieves this through *tile-based processing of vector elements*. In the SQL example above, one tile of input data is processed by each invocation of 'seq_window_agg_avg ()', which each produce one tile of transformed data that is passed to 'seq_sub ()' which, in turn, produces a tile and passes it as input to seq_cross (), and so on until the last function, seq_map () has exhausted its input. Transferring tiles between functions occurs entirely within CPU cache – data is not "materialized" back into main memory until fully transformed tiles emerge at the end of the pipeline.

Temporary Tables Impose Overhead

In contrast, to accomplish the task discussed above using a traditional SQL DBMS, the developer first creates three temporary tables:

```
CREATE TEMP TABLE mavg ( 5day float, 21day
float );
CREATE TEMP TABLE sub ( delta float );
CREATE TEMP TABLE crosses ( Price float );
```

The next step calculates 5-day and 21 moving averages and populates the temporary table 'mavg' with the results (this code assumes that the database system supports user-defined functions):

```
INSERT INTO mavg SELECT MovingAvg( Close, 5 ),
MovingAvg ( Close, 21 ) FROM Security WHERE
symbol = 'IBM';
```

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

*eXtreme*DB Financial Edition 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.

McObject LLC
33309 1st Way South
Suite A-208
Federal Way, WA
98003

Phone:+1 425 888 8505
info@mcobject.com
www.mcobject.com