

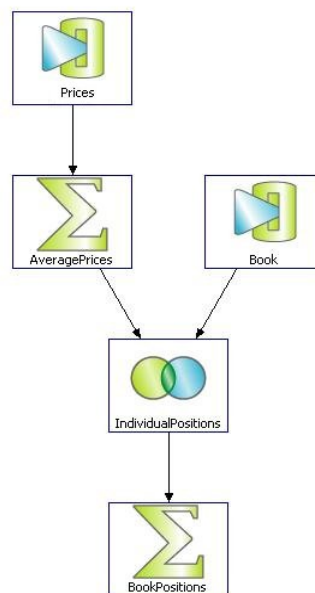


CEP and Storage – Where does all the data go?

By: Scott Kolodzieski, Architect

Introduction

Many CEP systems, regardless of the method of expressing the processing, may fundamentally be thought of as a directed graph of relational primitives (or stream operators). If one arranges the graph from top to bottom (in terms of data flow) one can think of data entering the top in the most raw form, and as the data works its way down the graph, being refined through further stages of processing. A simple example of this notion is given in the following graphical representation of a market feed driven price/positions model.



The input data in this simple example enters into the Prices and Book streams, while the other streams, AveragePrices, IndividualPositions and BookPositions are all computed based on the input data.

The question posed in this paper is, where is the data stored for these streams, and what options for storage are there?



Data Storage – Recoverability and Accessibility

In the Aleri Streaming Platform, each stream must be assigned to a store. Stores come in a variety of flavors, but essentially specify where (or if) data is stored and how it is indexed. The following table specifies the current storage types and indexing methods currently in use by the Aleri Streaming Platform.

<u>Store Type</u>	<u>Location of Data</u>	<u>Index Type</u>
Log	Disk	Tree
Memory	Memory	Tree or Hash
Stateless	no data retained	none

Recoverability (Log Store Technology)

If the application requires that all retained data is restored when it recovers state after a power failure or system shutdown, at least some of the data must be persisted to secondary storage. In the simple price positions example outlined above, the minimal subset of data that would have to be persisted to secondary storage is the two source streams. If all source streams of a model are stored in a Log Store, and thus persisted to disk, the rest of the model, i.e. all computed streams (these are called derived streams in the Aleri Platform), could be recomputed on system restart.

Depending on the amount of source data that is persisted, and on the complexity of all derived streams, the cost (in recovery time) of recomputing the model from the source data alone may be prohibitively expensive. In this case one could persist all streams to secondary storage by mapping both source and derived streams to a Log Store. The Log Store is periodically checkpointed, so recovery on restart is rapid.

One can see that the Aleri Streaming Platform offers two distinct modes for a CEP application when full recovery is required. One may map all source streams to a Log Store and allow the model to recompute all other streams on startup, or one may map all streams, both source and derived, to a Log Store and allow the checkpoint and fine-grained roll-forward recovery mechanism of the Log Store to fully restore state.

Accessibility

For data to be fully accessible, it must be stored and indexed. In the Aleri Streaming Platform, any stream that is stored in a Memory Store or Log Store may be accessed at any time via an ad-hoc SQL query, or by registering a subscription on the stream. The ad-hoc SQL query returns a rowset that indicates the current state of the stream, just as an SQL query on a normal relational database table would. The subscription mechanism, however is more dynamic. It returns the table state, like the SQL query, but continues to forward all changes to the stream as a set of keyed records (tagged with an insert, update or delete operation).



In the case of the simple example presented above, a full select statement on the final stream in the data model would yield a single row for each consolidated book position (select * from BookPositions). A subscription to this same stream would, like the select statement, return a single row for each consolidated book position; however this initial rowset would be followed by a series of records indicated how the consolidated positions are changing with new incoming prices.

A fundamental concept for both the SQL query and the subscription example is the base set of data for a stream at a given moment in time. The result of the ad-hoc query and the initial data set for a subscription is this base set of data. The fact that both a Memory Store and a Log Store each store the stream's records and index them, allow one to retrieve this base data for a stream. A stream that is mapped to a Memory Store or a Log Store maintain the state of their data. This allows the data in a stream to be searched, retrieved and presented to a client application.

On occasion, a stream's state is not important to the application. It may be a source stream that is never queried, or a derived stream that is really an intermediate stream in a chain of complex calculations. In both of these cases it is not important for the stream to store and index its data, but merely to take input and perhaps perform some calculations, then pass the (possibly modified) data along. This is truly the lightest weight use of a Stream in a CEP application. A highly optimized store, the Stateless Store, is designed for streams that fall into these categories. A stream that is mapped to a Stateless Store cannot be queried using an ad-hoc SQL statement as it does not contain any index data. A subscription to such a stream would always start with empty base data and would simply forward all new records that the stream produces.

Persistence, Throughput and Latency

It had been a common misconception that any CEP platform would have to be memory based in order to keep up with streaming data. Most database tuple stores are designed for rapid lookup of keyed data, but not designed for the rapid insertion (or updating) of vast amounts of streaming records. This is why, from it's inception, that a custom log-structured storage manager was developed and integrated as part of the Aleri Streaming Platform.

While certain CEP solutions depend on small time based windows that are self recovering, many CEP applications do not use windowed data to control the data size. The incoming data is bounded in the number of inserts, but unbounded in the number of updates and deletes. This sort of data yields an infinite stream of data that represents a finite keyed set of records which is constantly in flux. A natural example of such data is the current balance in a set of active managed accounts. These are constantly being updated, but the input data is not based on time or size based windows. Such applications, particularly in the financial and banking industries, require the persistence of streaming data along with guaranteed recoverability and consistency.



The Aleri Log Store technology, when coupled with modest SAN or SCSI raid hardware, streams and persists data to secondary storage at about twice the latency and with roughly half the throughput of Aleri's Memory Store. Absolute numbers are meaningless as it is clear that the number of streams, the complexity of the calculations, and the width (number of columns) in each stream, contribute to the throughput/latency profile of any CEP application. Nevertheless, in a modestly sized financial application we see upwards of hundreds of thousands of transactions being persisted to disk per second with single digit millisecond latencies.

The fact that the Aleri Streaming Platform allows base data to be mapped to Log Stores for full application recoverability, and all derived streams mapped to Memory Stores for minimal latency and maximal throughput, give the CEP application developer the unparalleled ability to develop an application that is capable of fully recovering from disk, yet fast enough to keep up with the fastest market data.

Conclusion

The Aleri Streaming Platform supports three storage modes, a persisted and fully indexed Log Store, a fully indexed Memory Store, and a light weight, non-indexed Stateless Store.

Each stream in a CEP application built must be mapped to a store – mapping the source streams, or all streams to one or more Log Stores allows for a fully state recoverable streaming application capable of running at market data feed rates. For applications that require no persistence, or that are self recoverable based on small time windows, a combination of Memory Stores and Stateless Stores allow for an ultra low latency CEP application capable of truly high event throughput.

UNITED STATES

Two Prudential Plaza, 41st Floor
Chicago, IL 60601
t +1 312 540 0100

430 Mountain Ave.
Murray Hill, NJ 07974
t +1 908 673 2400

UNITED KINGDOM

New Baltic House
65 Fenchurch Street
London EC3M 4BE
United Kingdom
t +44 (0) 20 7821 1110