DejaVu: A Complex Event Processing System for Pattern Matching over Live and Historical Data Streams

Nihal Dindar, Peter M. Fischer, Nesime Tatbul @ ETH Zurich

Motivation
- Find patterns on both live and archived data streams as well as detecting correlations among them
- Use cases: financial data analysis, healthcare monitoring, supply chain management, etc.

Goals
- Design and implement a CEP system that (i) detects and correlates patterns, (ii) works over both live and historical events, (iii) provides a uniform declarative query interface and (iv) scales to high throughput for high-volume streams

DejaVu Query Processing Engine
- Extends relational database engine MySQL by
  • pattern matching (semantic windows)
  • continuous query life cycle
- Pattern expressions composable with SQL
- Automata-based pattern computation
- Optimizations to reduce pattern matching cost
  • input sharing
  • state minimization
- Supports Pattern Correlation Queries (PCQs)
  • formal semantics
  • architectural extensions
  • cost model and optimizations

Optimizing PCQ Processing
- cost-model based optimizations, both architectural and algorithmic:
  • pattern computation before live-archive correlation
  • lazy archive pattern computation
  • recent input buffering
  • query result caching
  • join source ordering
- Throughput improvements up to 2 orders of magnitude

SQL-based Uniform Query Language
SELECT symbolL, initPriceL, minPriceL, initPriceA, ...
FROM StockLive
MATCH_RECOGNIZE
(PARTITION BY symbol
MEASURES A.symbol AS symbolL,
A.price AS initPriceL,
LAST(B.price) AS minPriceL
PATTERN(A B+)
DEFINE /* A matches any row */
B AS (B.price < PREV(B.Price)),
...
WHERE symbolA = symbolL
RECENCY = 7 seconds;

Upon detecting a fall in the current price of stock X on the live stream, look for a tick-shaped pattern for X within recent archive.

Performance on NYSE TAQ Data
Upon detecting a fall in the current price of stock X on the live stream, look for a tick-shaped pattern for X within recent archive.

DejaVu Client

Live Stream Storage
- In-memory storage engine for incoming streams
- Support for pull and push modes

Archived Stream Storage
- On-disk storage engine for archived streams
- Append-only, order-preserving, indexes

Recent Input Buffer
- Cache for efficient access to recent stream data
- Bulk inserts into archive stream storage

Query Result Cache
- Caches archive matches to avoid re-computation of archive patterns
- Significant performance benefits when recency correlation regions overlap
- Size at most linear to the size of the recency region (fits into memory in most cases)