A Performance Study of Event Processing Systems

Marcelo R.N. Mendes
Pedro Bizarro
Paulo Marques
(mnunes, bizarro, pmarques)@dei.uc.pt

First TPC Technology Conference on Performance Evaluation & Benchmarking:
TPCTC’09
Conventional BI Data Flow

- Seconds
- Minutes/Hours
- Hours/Days
Complex Event Processing (CEP) Overview

Event Processing Model
- Unbounded Event Streams;
- Events manipulated in main memory;

CEP system role:
- Filter/Correlate Events;
- Compute Aggregates;
- Detect Event Patterns;
- Identify/Predict Trends;
- Produce Alerts
Motivation

- Event Processing applications are usually time-critical;
- No standard benchmarks for CEP;
- Still little detailed performance information:
  - What are the performance bottlenecks?
  - Will performance degrade gracefully?
Performance Study
# Microbenchmarks

1. Selection and Projection
2. Aggregations / Windowing
3. Joins
4. Pattern Detection
5. Large Time-Based Windows
6. Adaptability
7. Multiple queries

- **Synthetic dataset;**

---

TPCTC: August, 2009  
CISUC, University of Coimbra - (m Nunes, bizarro, pmarques)@dei.uc.pt
Tests and Methodology

- 3 CEP systems were tested: Engines “X”, “Y” and “Z”;
- Tests consisted in:
  - A ramp-up phase (1 minute)
  - Measurement Interval (at least 10 minutes)
- One continuous query at a time
- Load Generation, Collection of Results:
  - FINCoS Framework (http://bicep.dei.uc.pt)
- Testbed:
  - HW: 1 server with 2 quad-core processors, 16GB RAM
  - SW: Windows 2008, x64 OS; Sun Hotspot x64 JVM.
Test Setup
Performance Results
1. Data Reduction

- Selection: A stream of events is filtered according to a given predicate;
  - Factor under Analysis: Predicate Selectivity
- Projection: Removal of attributes from events
  - Factor under Analysis: Number of Input attributes
- Metric: Throughput
1. Selection

**REMARKS:**
- Very-High Throughputs
- Limitations at client API's
2. Aggregations/Windowing

- Factors Under Analysis:
  - Aggregation Function: **AVG, MAX, STDEV, MEDIAN**
  - Window Definition:
    - Size;
    - Policy;
    - **EXAMPLE**: Keep the last 3 events from a given stream:

  Sliding Window: 1 2 3 5 6 7 ...

  Jumping Window: 1 2 3 5 6 7 ...

- Metric: Throughput
2. Aggregations/Windowing

- Performance Issues:
  - Maintenance of jumping windows on engine X;
  - Maintenance of sliding windows on engine Z;
  - Computation of $\text{MAX}$ function on engine Y.
3. Join Tests

1. **Window-to-Window**
   - Factors under Analysis: Window Size and Join Selectivity

2. **Stream to In-memory table**
   - Factor under Analysis: Table Size

3. **Stream to Database table**
   - Factor under Analysis: Table Size

- Join Selectivity: 100%;
- Number of Attributes:
  - Stream: 4
  - Table: 10
  - Output: 13
- Metric: Throughput.
3. Join Tests

CEP engine is responsible for maintaining the table in main memory and for performing the join.

Table is stored in an external database; data is retrieved through parameterized queries to DBMS.
6. Adaptability Tests

Steady Phase
(5 min)

Peak
(10 sec)

Recovery Phase
(5 min)

1,5 \times \lambda

\lambda
6. Adaptability Tests

- Definition of Adaptability Metrics:
  - Maximum Latency;
  - Latency Degradation;
  - Recovery time;

Results

<table>
<thead>
<tr>
<th></th>
<th>Engine</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>Max Latency</td>
<td>4.7 sec</td>
<td>1.3 sec</td>
<td>1.5 sec</td>
<td></td>
</tr>
<tr>
<td>Latency Degradation</td>
<td>×82.8</td>
<td>×57.4</td>
<td>×5.9</td>
<td></td>
</tr>
<tr>
<td>Recovery Time</td>
<td>43 sec</td>
<td>1.3 sec</td>
<td>1.5 sec</td>
<td></td>
</tr>
</tbody>
</table>
6. Adaptability Tests

Slowly recovering
7. Multiple Queries Tests

- 1st Test: Identical Queries
  - Goal: Assess Computation Sharing;
  - Metric: Throughput.

- 2nd Test: Overlapping Windows
  - Goal: Assess Memory Sharing;
  - Metric: Memory Consumption.

- Factor under analysis: Number of Queries.
7. Multiple Queries Tests

CONCLUSION: Only one engine showed evidences of implementing some kind of query plan sharing, but only for identical queries.
Final Remarks

- Aggregations and window policies: some surprises;
- Access to historical data might represent a bottleneck;
- Long GC pauses in CEP engines implemented in memory-managed languages hinder performance;
- Very different adaptability characteristics;
- None/Incipient Query Sharing;
- In General: still room for performance improvements.
Thanks!