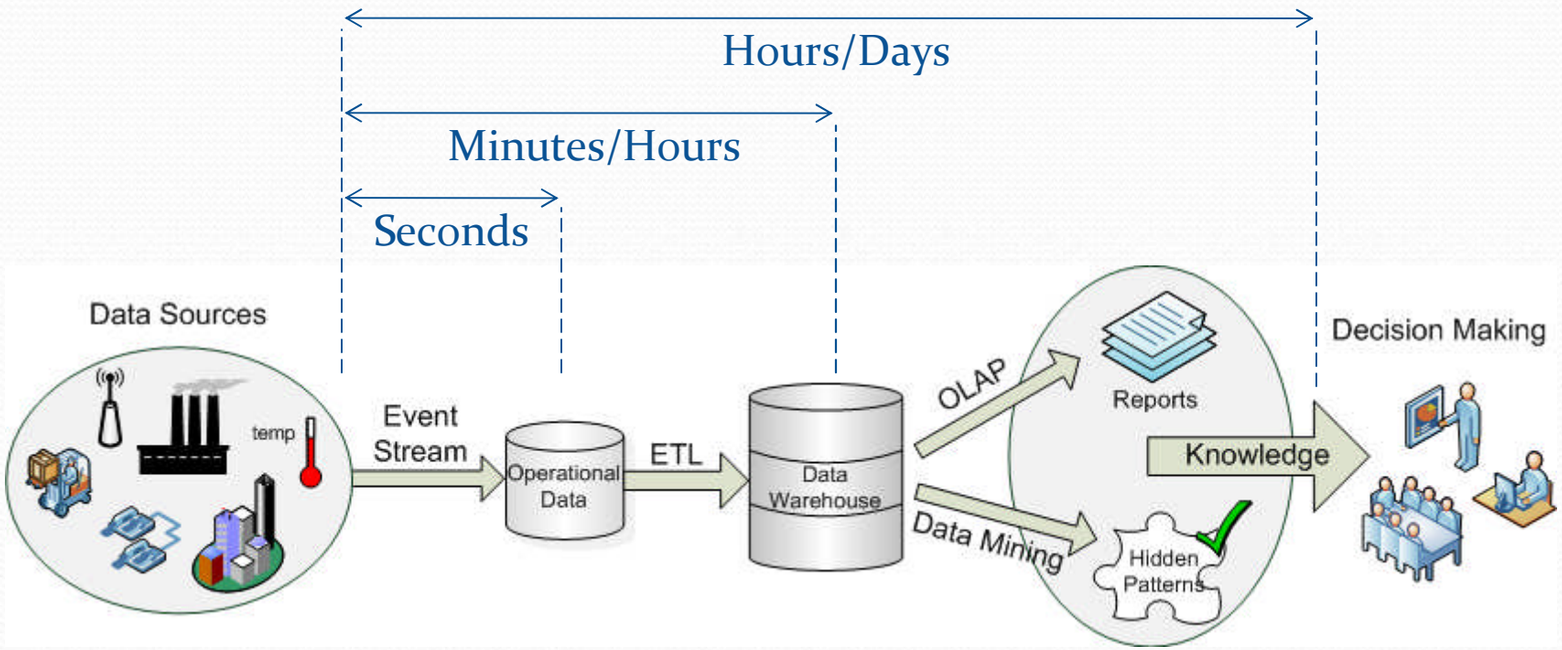
A Performance Study of Event Processing Systems

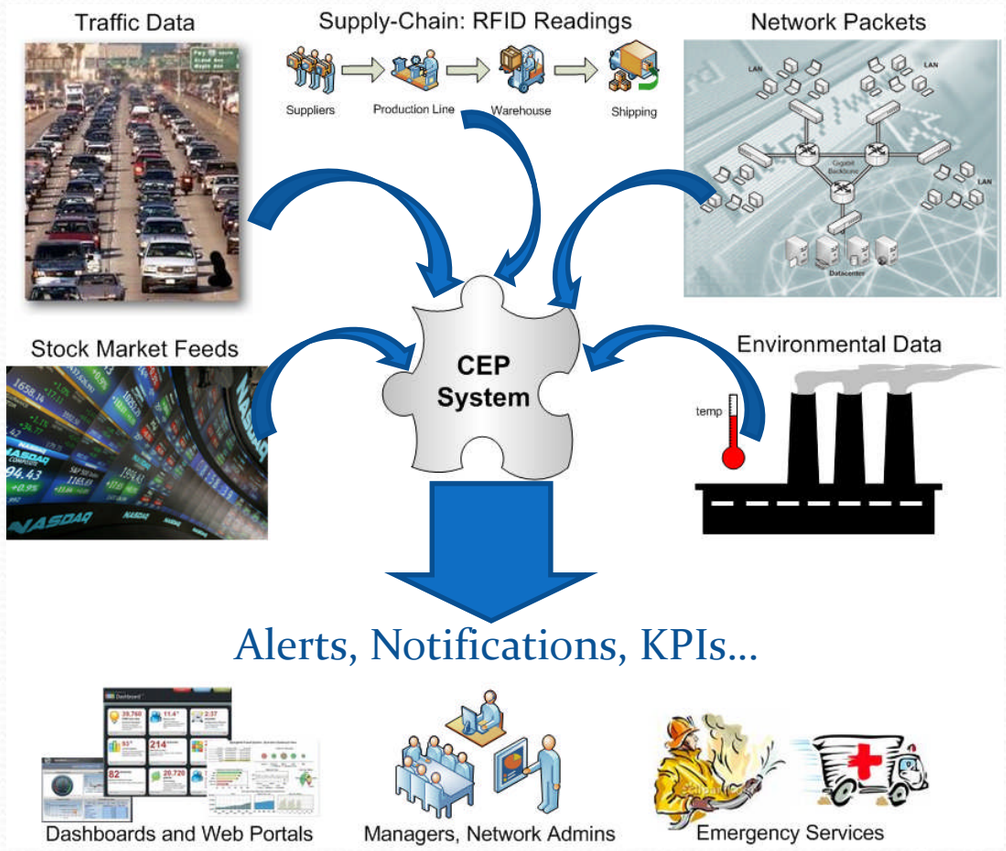
Marcelo R.N. Mendes
Pedro Bizarro
Paulo Marques
(mnunes, bizarro, pmarques)@dei.uc.pt

First TPC Technology Conference on Performance Evaluation & Benchmarking:
TPCTC'09

Conventional BI Data Flow



Complex Event Processing (CEP) Overview



Event Processing Model

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

CEP system role:

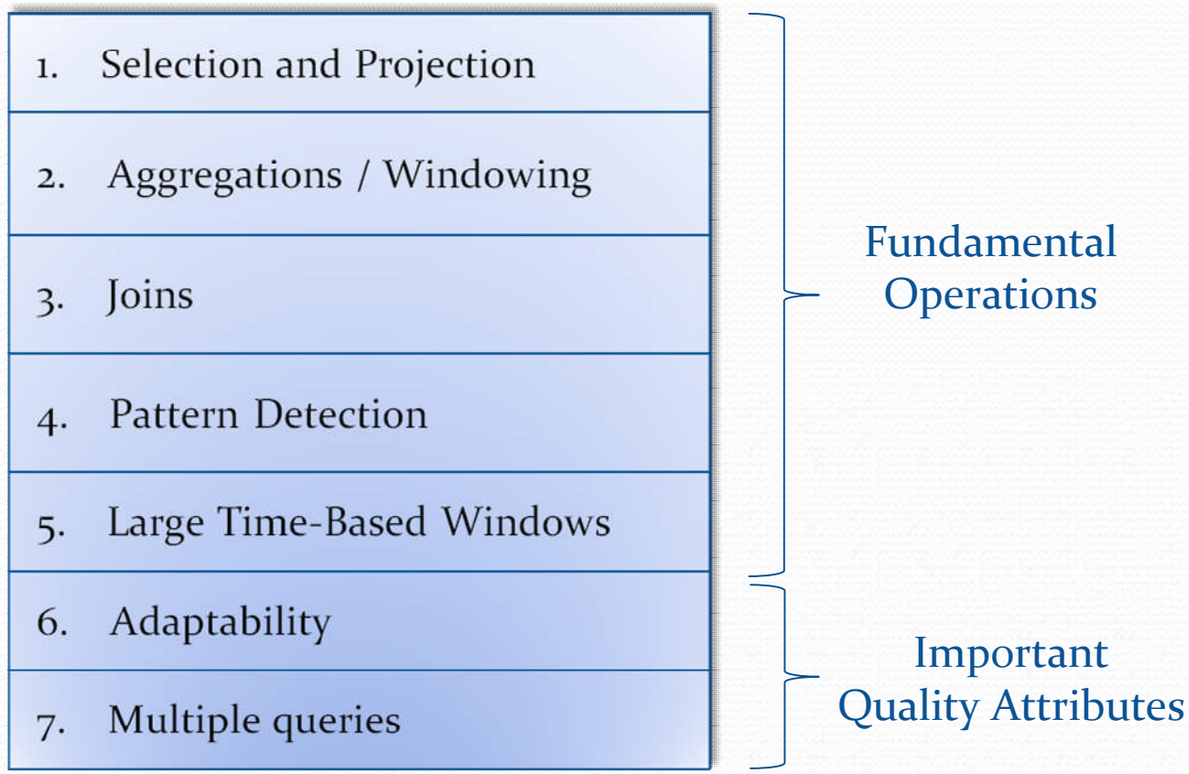
- Filter/Correlate Events;
- Compute Aggregates;
- Detect Event Patterns;
- Identify/Predict Trends;
- Produce Alerts

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

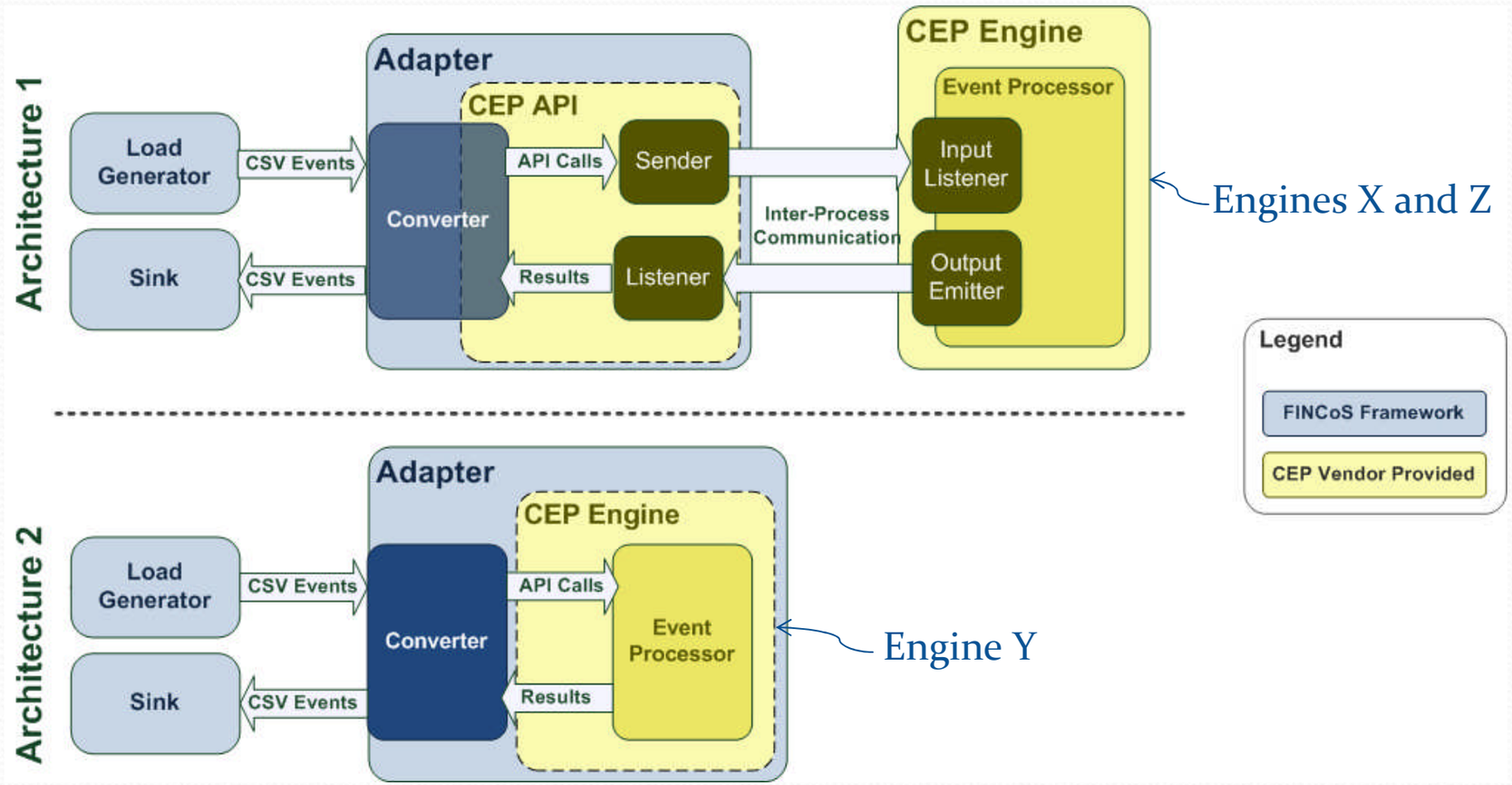


- Synthetic dataset;

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

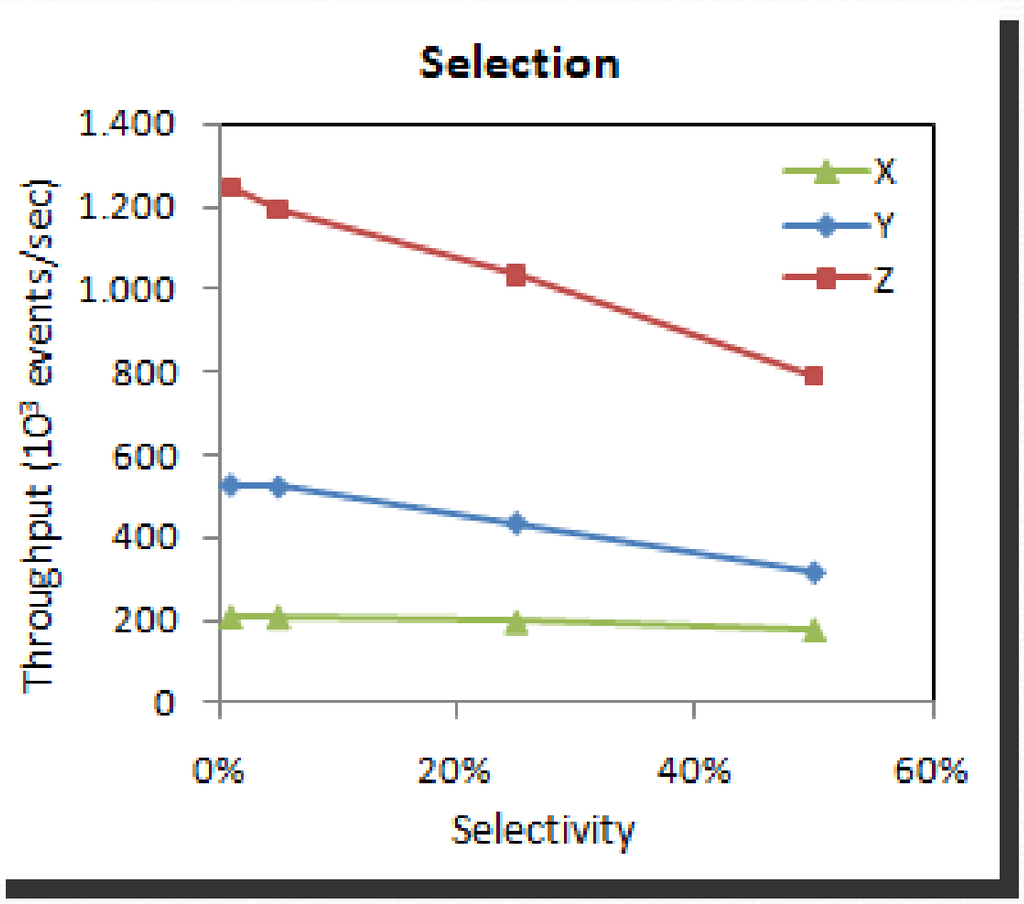

Architecture 1
Load Generator
CSV Events
Sink
CSV Events
Adapter
CEP API
Converter
API Calls
Sender
Results
Listener
Inter-Process Communication
CEP Engine
Event Processor
Input Listener
Output Emitter
Engines X and Z
Legend
FINCoS Framework
CEP Vendor Provided
Architecture 2
Load Generator
CSV Events
Sink
CSV Events
Adapter
CEP Engine
Converter
API Calls
Event Processor
Results
Engine Y

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

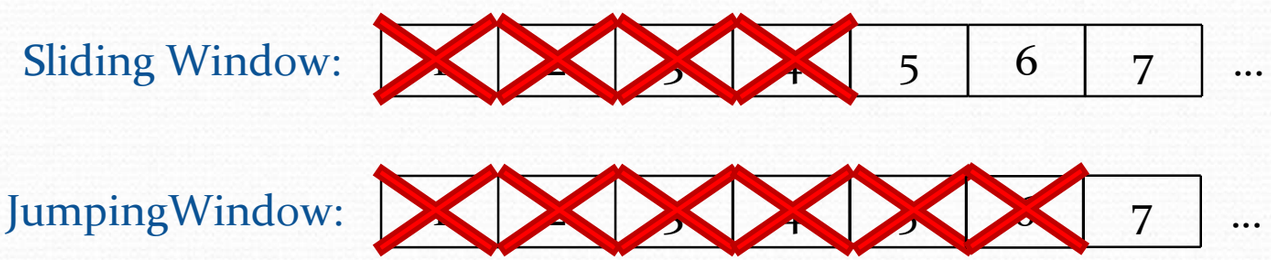
1. Selection



REMARKS:
- Very-High Throughputs
- Limitations at client API's

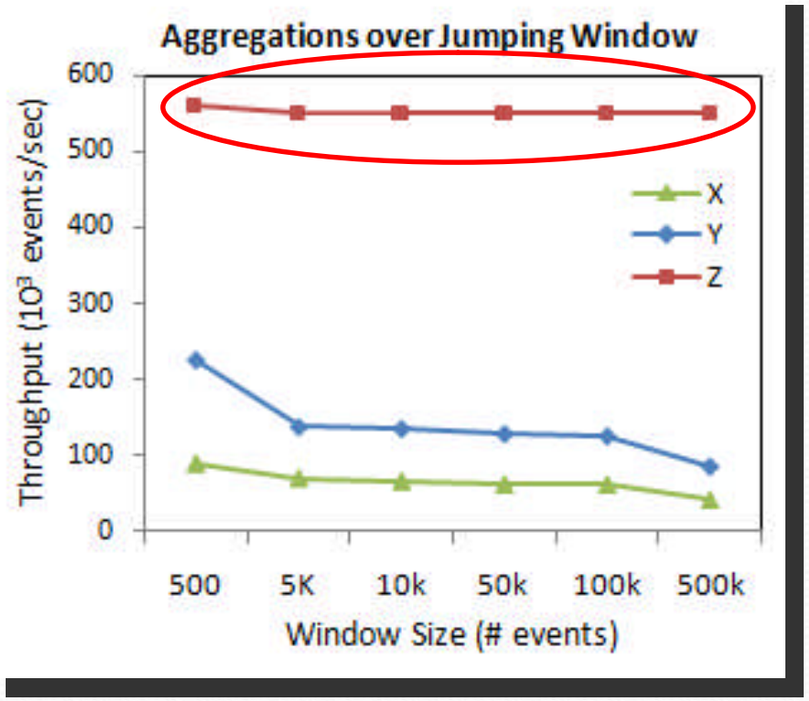
2. Aggregations/Windowing

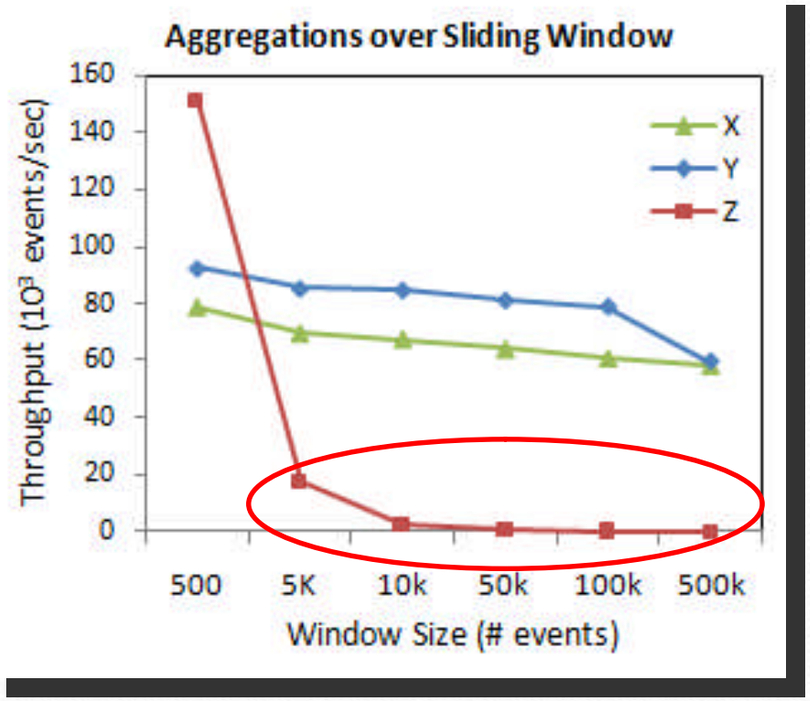
- Factors Under Analysis:
  - Aggregation Function: **`AVG`**, **`MAX`**, **`STDEV`**, **`MEDIAN`**
  - Window Definition:
    - Size;
    - Policy;
    - EXAMPLE: Keep the last 3 events from a given stream:



- Metric: Throughput

2. Aggregations/Windowing





- Performance Issues:
  - Maintenance of jumping windows on engine X;
  - Maintenance of sliding windows on engine Z;
  - Computation of 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

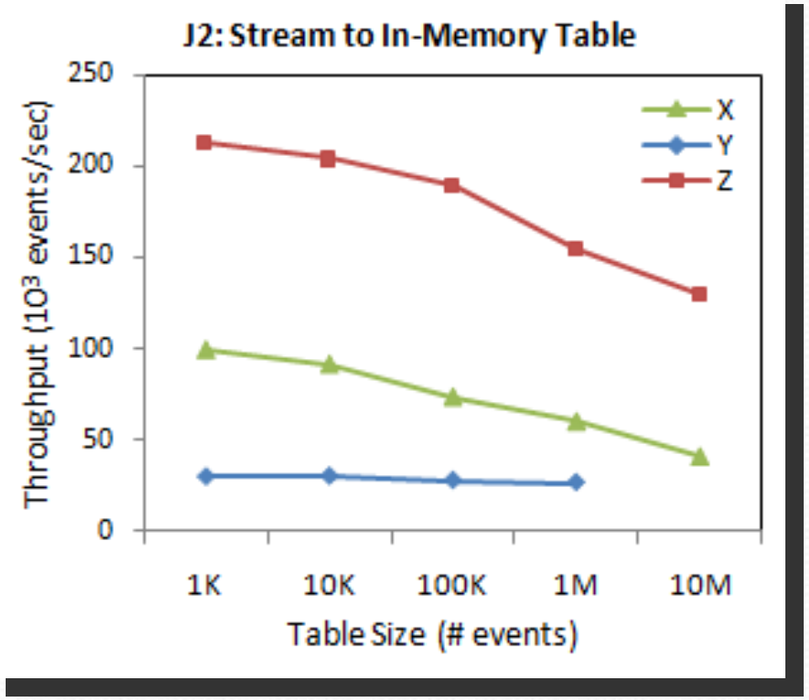
   -> Factor under Analysis: Table Size

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.

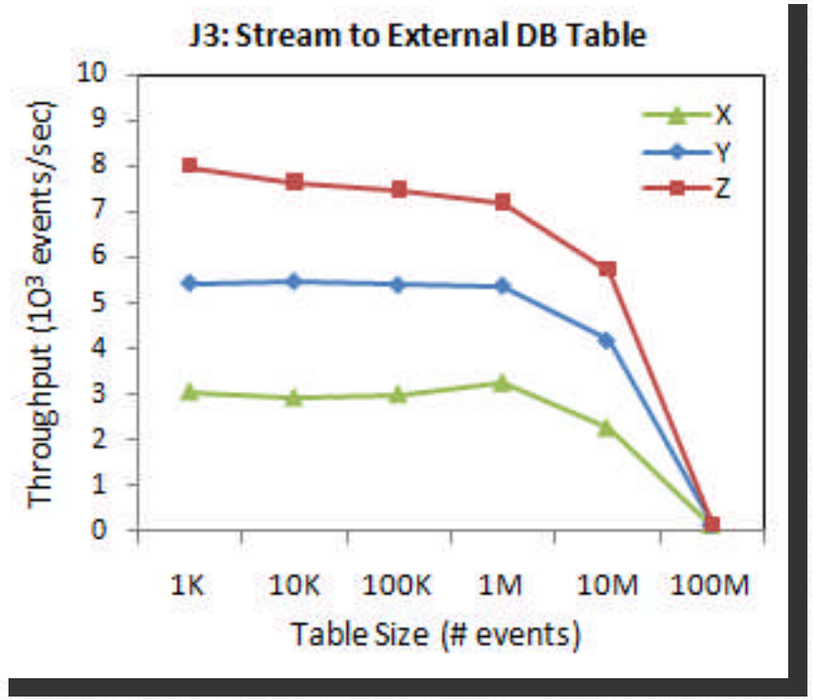
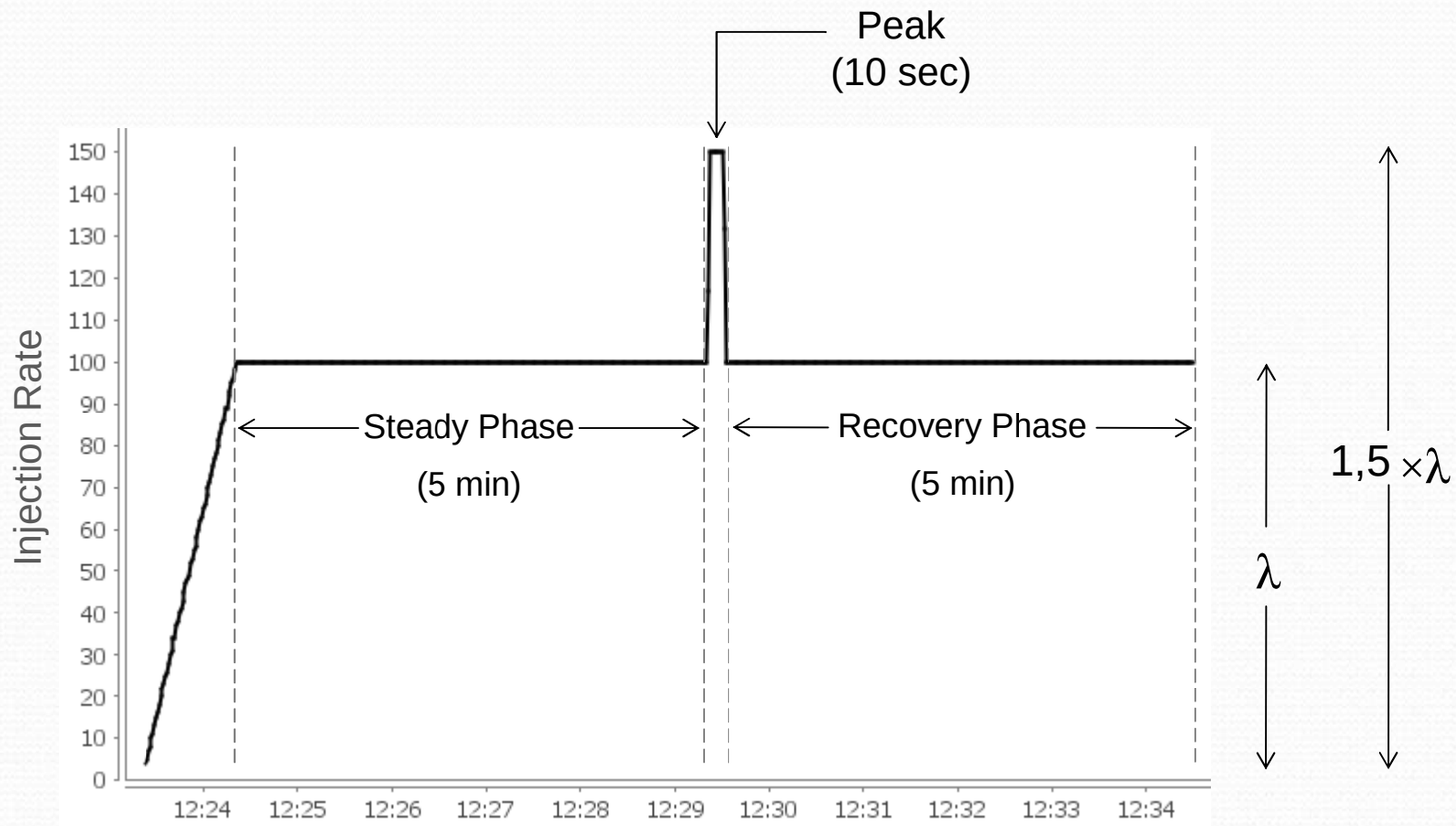


Table is stored in an external database; data is retrieved through parameterized queries to DBMS

6. Adaptability Tests



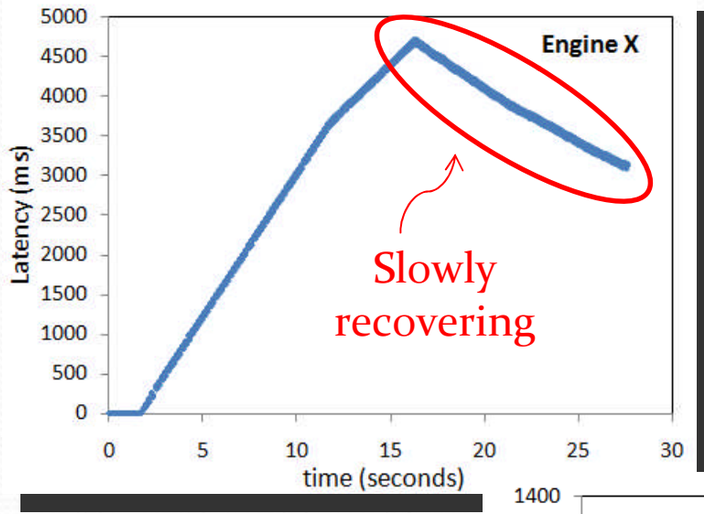
6. Adaptability Tests

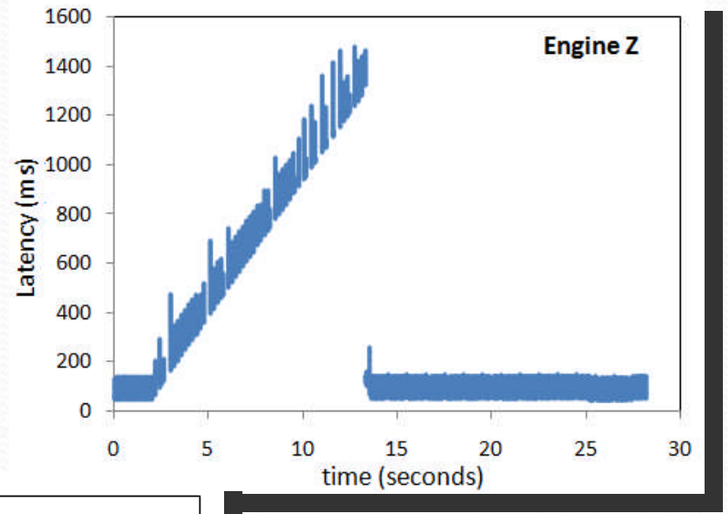
- Definition of Adaptability Metrics:
  - Maximum Latency;
  - Latency Degradation;
  - Recovery time;

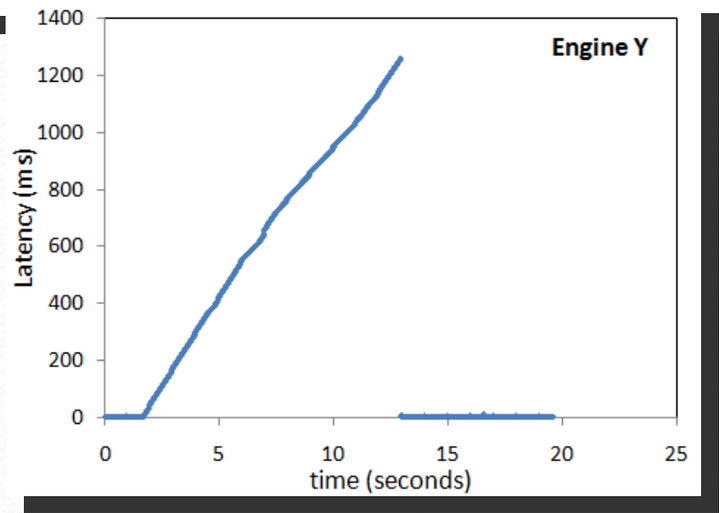
Results

	Engine		
Metric	X	Y	Z
Max Latency	4,7 sec	1,3 sec	1,5 sec
Latency Degradation	×82,8	×57,4	×5,9
Recovery Time	43 sec	1,3 sec	1,5 sec

6. Adaptability Tests



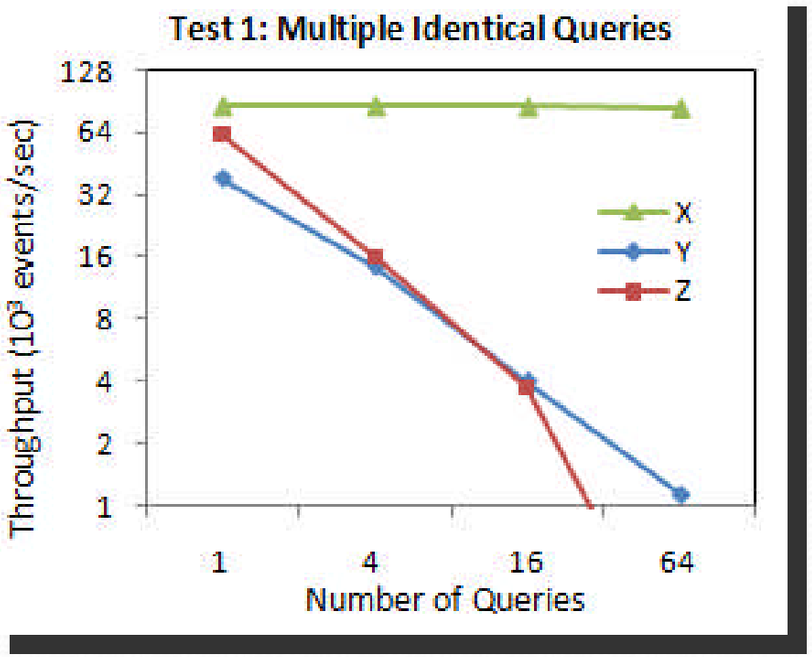


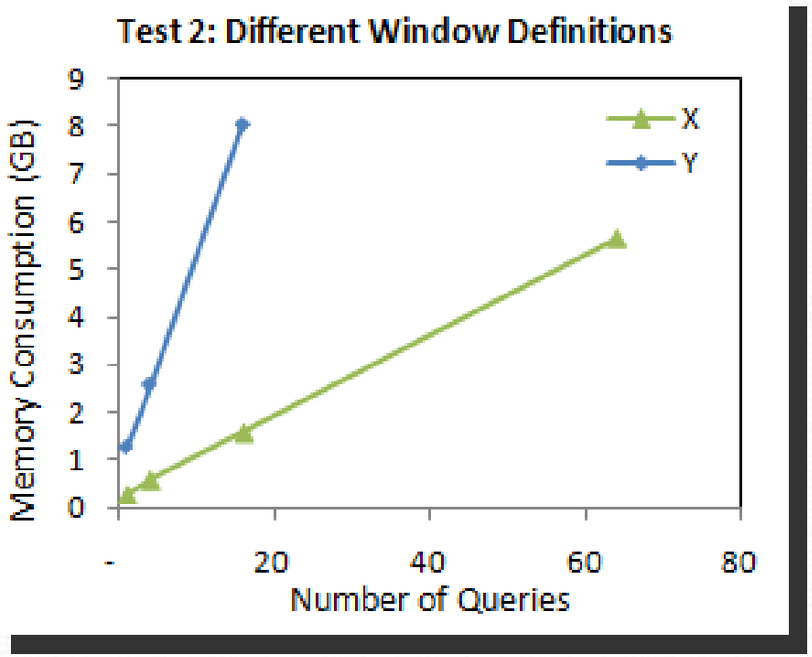


7. Multiple Queries Tests

- 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





<u>CONCLUSION</u>: Only one engine showed evidences of implementing some kind of query plan sharing, but only for <u>identical</u> 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!