Stream and Complex Event Processing
Benchmarking Information Flow Processing Systems

G. Cugola  E. Della Valle  A. Margara

Politecnico di Milano  Vrije Universiteit Amsterdam

cugola@elet.polimi.it  a.margara@vu.nl
dellavalle@elet.polimi.it
On Empirical evaluation of systems 1/2

• In 1995, "experimental evaluation in Computer Science: a quantitative study” by Tichy and collaborators [1] observed that
  • Computer scientist publish few paper with experimentally validated results
    • 40% no validation (12-15% in other fields)
  • In the papers that contains some evaluation, the fraction of the paper devoted to present it is small
    • 30% (70% in other fields)
• The lack of evaluation was a serious weakness in computer science in 1995
On Empirical evaluation of systems 2/2

• In 2009, "Empirical evaluation in Computer Science research published by ACM" by Wainer and collaborators [2], repeated the study in [1] for the paper published in 2005 and observed that
  • 70% of the paper are about design and modeling
    • 4% theory, 17% empirical, 4.7% hypothesis testing, and 3.4% other
  • Within the design and modeling class, 33% of the papers have no evaluation
  • Same results as 1995
• The lack of evaluation is still a serious weakness in computer science
Benchmarking databases

• The bible:
  • Database and Transaction Processing Performance Handbook by Jim Gray [3]

• The question:
  • What system does the job with the lowest cost-of-ownership?

• The solution
  1. **Define a benchmark** (or workload)
  2. **run** on several different systems
  3. **measure**
     • the **performance**: a throughput metric (work/second)
     • the **price**: a five-year cost-of-ownership metric
BENCHMARKING DATABASES

THE NEED FOR DOMAIN-SPECIFIC BENCHMARKS

• Generic benchmarks give some sense of the relative performance and price/performance of a system.
• No single metric can measure the performance of computer systems on all applications.
• Domain-specific benchmarks are a response to diversity of computer system use.
  • Each benchmark specifies a synthetic workload characterizing typical applications in that problem domain.
  • The performance of this workload on various computer systems then gives a rough estimate of their relative performance on that problem domain.
Benchmarking databases

The Key Criteria For a Domain-Specific Benchmark

• Relevant
  • measures performance and price/performance of systems when performing typical operations within the problem domain

• Portable
  • easy to implement on many different systems

• Scalable
  • applies to small and large computer systems

• Simple
  • understandable, otherwise it will lack credibility.
Benchmarking databases

Dimension to be benchmarked

• Complex Queries
• Complex Transactions
• Utility Operations
  • e.g., building indexes, altering tables
• Mixed Workloads
Benchmarking databases

Data base benchmarks in [3]  1/2

• TPC BM™ A/B/C
  • Online Transaction Processing

• Wisconsin
  • Relational Queries

• AS³AP
  • Mixed Workload of Transactions, Relational Queries, and Utility Functions

• Set Query Benchmark:
  • Complex Queries and Reporting

• Engineering Database Benchmark:
  • Engineering Workstation-Server
## Benchmarking databases

### Data base benchmarks in [3]

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Complex Queries</th>
<th>Complex Transactions</th>
<th>Utility Operations</th>
<th>Mixed Workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC BM™ A/B/C</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wisconsin</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS³AP</td>
<td>x</td>
<td>Simple only</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Set Query Benchmark</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineering Database</td>
<td>simple</td>
<td>simple</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

9
Benchmarking databases

How to Use Those Benchmarks

• compare different software products on one machine
• compare different software products on one machine
• compare different machines in a compatible family
• compare different releases of a product on one machine
Benchmarking databases

Benefits of benchmarking

- make competing products comparable
- accelerate progress, make technology viable

• Different properties
• New challenges
• New key performance indicators
• New stress tests
What about DSMS/CEP/SR?

Different properties

• Time Model
  • Implicit vs. explicit

• Time Semantics
  • Punctual vs. interval

• Query Model
  • Stream analysis vs. event pattern matching

• Quality of Service
  • Best effort vs. guaranteed QoS

• Distribution

• Background Data support

• Inference Support
What about DSMS/CEP/SR?

New challenges

• Time Modeling
• Querying
• Managing Bursts
• Managing Background Data
• Inference Expressivity
What about DSMS/CEP/SR?

New challenges - Time Modeling

• Choosing a specific model – and a corresponding semantics – for representing time can significantly impact the performance of the system
What about DSMS/CEP/SR?

New challenges - Querying

• Choosing an appropriate strategy for storing, accessing, and discarding partial results
• Choosing operators that determine the scope of processing
• Choosing mechanism for triggering queries and the management of multiple queries
What about DSMS/CEP/SR?

New challenges - Managing Bursts

• Scarify completeness and correctness vs. scale-out and scale-in back
What about DSMS/CEP/SR?

New challenges - Managing Background Data

• If the background knowledge is large or pulling it on demand is time consuming it can become a bottleneck for real-time processing

• Large background data may be difficult to distribute
What about DSMS/CEP/SR?

New challenges – Inference Expressivity

• Choosing the right expressivity
  • ++expressivity $\rightarrow$ ++processing time

• Temporal reasoning vs. built-in time-management
What about DSMS/CEP/SR?

Properties and challenges

<table>
<thead>
<tr>
<th></th>
<th>Time Modeling</th>
<th>Querying</th>
<th>Managing Bursts</th>
<th>Managing Background Data</th>
<th>Inference Expressivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Model</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Time Semantics</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Query Model</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Quality of Service</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Distribution</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Background Data</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Inference Support</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
What about DSMS/CEP/SR?

Key performance indicators (KPIs)

• DSMS/CEP/SR are reactive systems
  • Answer must arrive within a given time
  • Answers received after that time are useless

• KPIs
  • Response time
    • Average/$x^{th}$ Percentile/Minimum/Maximum
  • Maximum input throughput
  • Time to accuracy
    • hopefully equal to response time
  • Time to completion
    • not necessarily equal to response time
  • Minimize Resource utilization
    • RAM, bandwidth
What about DSMS/CEP/SR?

What to benchmark for

• Load Balancing
• Filter, Joins (and Inference) on Flow Data Only
• Filter, Joins and Inference in Flow and Background Data
• Aggregates
• Unexpected Data (out of order, and noisy)
• Schema size and expressiveness
• Changes in Background-Data
What about DSMS/CEP/SR?

Existing benchmarks

• DSMS
  • Linear Road Benchmark [5]
    • http://www.cs.brandeis.edu/~linearroad/

• CEP
  • Fast Flower Delivery [6]
    • http://www.ep-ts.com/content/view/80

• Stream Reasoners (I'd better to say RDF stream processor)
  • SR-Bench [7]
    • http://www.w3.org/wiki/SRBench
  • LS-Bench [8]
    • http://code.google.com/p/lsbench/
Existing benchmarks

Linear Road Benchmark – The idea

- Simulates an unban highway system that uses ‘variable tolling’ (i.e., congestion-based pricing).
Existing benchmarks

Linear Road Benchmark – The Challenges

• Semantically Valid Input
  • Use a simulator: MIT Traffic Simulator

• Many Correct Results
  • continuous queries results may depend upon evolving historical state or the arrival order tuples on a stream, and therefore several different results for the same query may be “correct”

• No Query Language
  • Queries are language-agnostic, yet have a clear semantics
Existing benchmarks

Linear Road Benchmark – The data

• A 3-hour, single expressway worth of input data consists of
  • 12 million position reports
  • 67000 account balance
  • 14000 daily expenditure query requests.

• As the simulation time progresses, the input data exhibits a monotonically increasing distribution with time
  • from 15 records per second to 1700 records per second.
Existing benchmarks

**Linear Road Benchmark – The queries**

- **Two continuous queries**
  - Calculates a segment toll every time a vehicle enters the segment.
  - Detects and reports accidents and adjusts tolls accordingly.

- **Three Historical queries**
  - Request an account balance
  - Day’s total expenditure for a given vehicle
  - Prediction of travel time between two segments using historical data

- **KPI**
  - Each of these queries must be answered with a specified accuracy and within a specified response time.
Existing benchmarks

Fast Flower Delivery – The idea

• The flower stores association in a large city has established an agreement with local independent van drivers to deliver flowers from the city's flower stores to their destinations.

• The CEP-centric system is used to support this business need.

• No KPIs

• More a use case than a benchmark
Existing benchmarks

Fast Flower Delivery – The idea

2/2
1. Bid phase
   - When a store gets a flower delivery order, it creates a request, which is broadcasted to relevant drivers within a certain distance from the store, with the time for pick up (typically now) and the required delivery time if it is an urgent delivery.

2. Assignment phase
   - A driver is then assigned and the customer is notified that a delivery has been scheduled.
   - Assignment can be both automatic (see phase) or manually performed by the store.
3. Delivery process
   • The driver picks up the delivery and delivers it, and then person receiving the flowers confirms the delivery time by signing for it on the driver's mobile device.

4. Ranking Evaluation
   • The system maintains a ranking of each individual driver based on his or her ability to deliver flowers on time.

5. Activity monitoring
   • Creates reports on driver's performances
Existing benchmarks

Fast Flower Delivery – Additional info

• Each store has a profile that can include a constraint on the ranking of its drivers, for example a store can require its driver to have a ranking greater than 10. The profile also indicates whether the store wants the system to assign drivers automatically, or whether it wants to receive several applications and then make its own choice.
Existing benchmarks

SRbench - Challenges

• Proper benchmark dataset
  • use real-world datasets from LOD
    • LinkedSensorData

• No standard query language:
  • natural language query definition and
  • three implementations
  • SPARQLStream
  • CQELS
  • C-SPARQL
Existing benchmarks

SRbench - Datasets

• Use case: weather information application
• Data model

• Data:
  • 10,000 weather station
  • 5 sensor in each: temperature, visibility, precipitation, pressure, wind speed and humidity
  • Millions of sensor observations
### Existing benchmarks

**SRBench - Queries**

<table>
<thead>
<tr>
<th>Addressed features per query</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
<th>Q11</th>
<th>Q12</th>
<th>Q13</th>
<th>Q14</th>
<th>Q15</th>
<th>Q16</th>
<th>Q17</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 query form</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 reasoning</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 CQL feature</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T,</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>7 data access</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,S</td>
<td>0,S</td>
<td>0,S</td>
<td>0,S</td>
<td>0,S,F</td>
<td>0,S,G</td>
<td>0,S,G</td>
<td>0,S,D</td>
</tr>
</tbody>
</table>

Existing benchmarks

SRbench - KPI

• Feature coverage
• Correctness
  • In case of approximate answers: precision and recall
• Throughput
• Scalability
• Response Time
Existing benchmarks

LS-Bench – Use case

• Social Network + Social Streams
Existing benchmarks

LS-Bench – Data set

• Synthetic data generation
  • http://lsbench.googlecode.com/files/sibStream0617.tar

• It resembles reality
  • In the social network form and dynamics
  • In the skewed distribution of posts/comments
  • In GPS tracks

• Customizable
  • Duration
  • Maximum number of posts/comments/photos for each user per week
  • Correlation probabilities
## Existing benchmarks

### LS-Bench – Queries

<table>
<thead>
<tr>
<th>Patterns covered</th>
<th>S</th>
<th>$N_P$</th>
<th>$N_S$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>J</td>
<td>A</td>
</tr>
<tr>
<td>$Q_1$</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_2$</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_3$</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_4$</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$Q_5$</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_6$</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_7$</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$Q_8$</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$Q_9$</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$Q_{10}$</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_{11}$</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$Q_{12}$</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

### Legend

- **F**: Filter
- **J**: join
- **A**: aggregation
- **E**: nested query
- **N**: negation
- **U**: union
- **T**: top-k
- **S**: use static data

$N_P$: number of patterns

$N_S$: number of streams
Existing benchmarks

LS-Bench – KPI

• Feature coverage

• Correctness
  • by comparison of results of different implementations 😞

• Performance: input throughput
## Existing benchmarks

### Evaluation of existing benchmarks [4]

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Linear Road Benchmark</th>
<th>Fast Flower Delivery</th>
<th>SR-bench</th>
<th>LS-Bench</th>
</tr>
</thead>
<tbody>
<tr>
<td>load balancing</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Filter, Joins (and Inference) on Flow Data Only</td>
<td>yes (no)</td>
<td>yes (no)</td>
<td>yes (yes)</td>
<td>yes (no)</td>
</tr>
<tr>
<td>Filter, Joins (and Inference) in Flow and Background Data</td>
<td>yes (no)</td>
<td>yes (no)</td>
<td>yes (no)</td>
<td>yes (no)</td>
</tr>
<tr>
<td>Aggregates</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Unexpected Data</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Schema size expressiveness</td>
<td>yes(no)</td>
<td>yes(no)</td>
<td>yes(no)</td>
<td>yes(no)</td>
</tr>
<tr>
<td>Changes in Background-Data</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
References


