Grand Challenge: The TechniBall System

Avigdor Gal, Sarah Keren, Mor Sondak, Matthias Weidlich, Hendrik Blom, Christian Bockermann

Technion - Israel Institute of Technology, Haifa, Israel
TU Dortmund, Germany
What did we use for prototyping?
Streams

Java-based framework

Data flow defined in XML

Compiled into computation graph for Streams or Storm

Basic elements: sources, sinks, processors

```java
public interface Processor {
    public Data process(Data input);
}
```
Esper

Open source CEP engine

Well-documented and optimized over years

Queries are defined in EQL, a declarative, SQL like language

- Powerful: aggregations, event context definitions
- Programmable: Java code can be injected in the query

```sql
select avg(price) from StockTick.win:time(30 sec) where symbol='IBM'
```
Integration

An Esper processor for Streams

– Direct integration of Esper query in the Streams data flow
– Configuration of type mapping for Esper
– Configuration of Streams process to which results shall be emitted

<stream.esper.Query condition="" output="R">
  Esper query
</stream.esper.Query>
How did we implement it?
**TechniBall Architecture**

The Soccer Stream

<table>
<thead>
<tr>
<th>Event</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>id, ts, x, y, z, v, vx, vy, vz, ax, ay, az</td>
</tr>
<tr>
<td>E1</td>
<td>E0 + pid, active, pos.x, pos.y, pos.z, [player.x, player.y]</td>
</tr>
<tr>
<td>E3</td>
<td>E1 + intensity</td>
</tr>
</tbody>
</table>

Optional GameView Processor (Figure 11)
<?xml version="1.0" encoding="UTF-8"?>
<container import="stream.soccer">
  <properties> ... </properties>
  <stream id="soccerData"
    class="stream.io.FastSocStream" url="file:/path" />

  <process input="soccerData">
    <stream.soccer.AddMetaData />
    <stream.soccer.AddGameIsActive />
    <stream.soccer.AddPosition />
    <Enqueue queue="Q:1" condition="{%data.pid} > 0" />
    <Enqueue queues="Q:2,Q:3" />
  </process>

  <process input="Q:1">...</process>
  <process input="Q:2">...</process>
  <process input="Q:3">...</process>
  <process input="Q:4">...</process>
  <process input="R"> <stream.io.JSONOutput />
  </process>
</container>
Example Running Analysis

```xml
<process input="Q:1">
  <stream.soccer.AddIntensity />
  <stream.esper.Query condition="%{data.pid} @gt 0" output="R"> SELECT pid AS player,
  sum(case when int=1 then distance else 0 end)as trot_distance,
  sum(case when int=1 then (ts_stop-ts_start)/1000000000 else 0 end)as trot_time
  FROM Data.win:time(60000 sec) GROUP BY pid
</stream.esper.Query>
</process>

<process input="Q:1">
  <stream.soccer.AddIntensity />
  <stream.soccer.AggregateCurrentIntensity output="R" />
</process>
```
How does it perform?
Streams vs. Esper (Q3)
Throughput

# Events / Second

- Query 1
- Query 2
- Query 3
- Query 4
Latency

Nanoseconds

query 1  query 2  query 3  query 4
Throughput with Parallelization

- **Standard setup (4 queues)**
- **Single-threaded (single queue)**
- **Split (two queues)**

<table>
<thead>
<tr>
<th># Events / Second</th>
<th>Standard setup</th>
<th>Single-threaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>150,000</td>
<td>q1+q3</td>
<td></td>
</tr>
<tr>
<td>250,000</td>
<td>q1+q3</td>
<td></td>
</tr>
</tbody>
</table>
Visualization
Conclusion

TechniBall as an innovative and efficient solution to the DEBS Grand Challenge

Interplay of a high-level CEP and low-level event processing
  – Trade-off between understandability, maintainability, performance requirements
  – Final submission was purely Streams-based
  – Esper implementation valuable for understanding the queries

Further details: http://jwall.org/TechniBall/