Grand Challenge: SPRINT Stream Processing Engine as a Solution

Yingjun Wu\textsuperscript{1}, David Maier\textsuperscript{1,2}, Kian-Lee Tan\textsuperscript{1}

\textsuperscript{1}National University of Singapore
\textsuperscript{2}Portland State University
Outline

- Processing Challenge Queries
- System Architecture
- Evaluation
Query 1: Running Analysis

Sensor-generated instant velocity does not directly reflect players’ real running speeds!

Figure: Sensor-generated velocity streams.

Figure: Edward Muybridge's 1887 motion study of an athlete running.*

Processing Challenge Queries
Query 1: Running Analysis

• Observation: average velocity stream is not stable enough for reliable measurement.

• Strategy:
  • Every running status with duration less than 0.1 second will be removed as noise;
  • Additional cross-status sections are inserted between each sibling-status pair.
Processing Challenge Queries

Query 1: Running Analysis

Generate reliable measurements:

- Employ a special data structure: Frame.
- Fields: <start_time, update_time, end_time, status>

![Diagram showing data structure and measurements]
Processing Challenge Queries

Query 1: Running Analysis

Example of compressed linked list

```
start  0.0s  1.3s  5.5s  6.9s  10.0s
end    standing  trot  medium  trot
status 10.0s  12.3s  14.1s  15.9s  17.1s
medium  sprint  low  trot
```
Processing Challenge Queries
Query 2: Ball Possession

Key point: detect ball-hit events with velocity-based approach.

Figure: Ball acceleration and velocity.
Processing Challenge Queries
Query 2: Ball Possession

Velocity-based detection of ball-hit events:

- Continuously detect “sudden change” event;
  - “sudden change”: a change of +5 or -2 speed units in 0.0015 second.
- Adopt approximate NN search for nearest player;
- Use the “blind eye” method to improve efficiency.
  - Once the ball-hit event is confirmed, SPRINT pauses velocity monitoring for 0.5 seconds.
Processing Challenge Queries

Query 2: Ball Possession

Maintain a single list to keep track of the possession-exchange event.

- The two teams control the ball alternately during the game.

![Diagram showing ball possession exchange between Team 1 and Team 2 with specific times.]
Processing Challenge Queries

Query 3: Heat Map

- Simple approach (window-based):
  - Partition field into 6400 cells;
  - Record how long each player spent in each cell;
  - Calculate output by summing up values from corresponding cells.

- Optimized approach (frame-based):
  - Use frame-based sliding windows;
  - Three fields: <Cell ID, enter-cell timestamp, leave-cell timestamp>.
Processing Challenge Queries
Query 3: Heat Map

Simple illustration with 4 cells

Figure: A player p is “wandering” over 4 cells.
Processing Challenge Queries
Query 4: Shot on Goal

The procedure of detecting goal attempts

Is ball hit?
Y
Can reach baseline?
N Not an attempt
  Y
Can reach goal area?
N Not an attempt
  Y
An attempt
System Architecture

- Centralized stream processing engine
  - Implemented in C++.

- Innovative aspects:
  - Lock-free ring buffer;
  - Frame-based sliding windows;
  - Parallel computation;
  - Dynamic adaptation.
SPRINT System Architecture

Data Sources → Preprocessor → Ring Buffer → Parallel query processors
System Architecture
Lock-free ring buffer

- Adopt a one-producer-multiple-consumer model to handle incoming stream data.
- Follow the lock-free ring buffer model proposed in LMAX Disruptor;
  - Adopt CAS locks, introduced in C++ 11.
- Set the message size to 4 KB to maximize memory bandwidth.
System Architecture
Frame-based sliding windows

- Use a linked list of frames to compress the content of sliding windows by only tracking status transitions;
- Adjacent panes with same status are merged to eliminate redundant information;
- Maintain 3 basic fields: <start_time, end_time, status>.
  - Additional fields are also allowed!

Figure: the merging procedure of sibling frames.
System Architecture
Parallel computation

Two levels of multi-core computation:

- Inter-query level: Benefit from ring buffer.
- Intra-query level: Query 1 and Query 3 can be easily parallelized.

Figure: Intra-query level multi-core computation.
System Architecture
Dynamic adaptation

Load shedding strategies

Incoming data → Shared Ring Buffer → Query processing

- Load shedder position #1 controls global load shedding rate
- Load shedder position #2 controls local load shedding rate
Evaluation

• Experiments conducted on a single machine
  • Four 2.00 GHz Intel processors
  • 2 GB memory
  • CentOS 5.8

• Metrics
  • Throughput
  • Precision and recall
Evaluation

Throughput

- Definition: number of input events consumed per second;
- Run each query separately to judge single-query performance;
- Run all queries simultaneously to test overall system performance.
Evaluation

Throughput as input events per second

![Graph showing throughput as a function of load shedding rate for different queries. The x-axis represents the load shedding rate (0% to 100%), and the y-axis represents throughput (thousand events/second). Different colored lines represent different queries, with Query 1 having the highest throughput and Query 4 having the lowest. The overall throughput is also shown as a blue line.]
Evaluation

Precision and recall

- Verify the correctness of Query 2 and Query 4 by comparing with the provided referee events;
- Set load-shedding rate to different values.
## Evaluation

### Precision and recall

<table>
<thead>
<tr>
<th>Query</th>
<th>Load-shedding</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0%</td>
<td>93.3%</td>
<td>90.8%</td>
<td>92.0%</td>
</tr>
<tr>
<td>Query 2</td>
<td>25%</td>
<td>93.3%</td>
<td>90.8%</td>
<td>92.0%</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>90.1%</td>
<td>87.9%</td>
<td>88.9%</td>
</tr>
<tr>
<td></td>
<td>75%</td>
<td>91.5%</td>
<td>75.1%</td>
<td>82.5%</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Query 4</td>
<td>0%</td>
<td>84.2%</td>
<td>79.6%</td>
<td>82.0%</td>
</tr>
<tr>
<td></td>
<td>25%</td>
<td>84.2%</td>
<td>79.6%</td>
<td>82.0%</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>83.3%</td>
<td>72.1%</td>
<td>77.3%</td>
</tr>
<tr>
<td></td>
<td>75%</td>
<td>81.8%</td>
<td>62.8%</td>
<td>71.1%</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>N/A</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>
Thank you!