Optimized On-Demand Data Streaming from Sensor Nodes

Jonas Traub, Sebastian Breß, Asterios Katsifodimos, Tilmann Rabl, Volker Markl
About me

Jonas Traub

jon@s-traub.com
Jonas.traub@tu-berlin.de

• Researcher and PhD candidate at
  – Technische Universität Berlin (DIMA)
  – German Research Center for Artificial Intelligence (DFKI) / (IAM)

• Working with Volker Markl

• Before
  – Master’s degree in Computer Science (KTH Stockholm and TU Belin)
  – Bachelor’s degree in Applied Computer Science (DHBW Stuttgart)
  – Four years at IBM in Germany and the USA
The Sensor Cloud

Sensor Nodes

Sensor Cloud

Stream Analysis Cluster

Real-time insights
Billions of sensor nodes form a sensor cloud and provide data streams to analysis systems.
Billions of sensor nodes form a sensor cloud and provide data streams to analysis systems.
Billions of sensor nodes form a sensor cloud and provide data streams to analysis systems.

The Sensor Cloud

Produce / Read all possible data   Transfer all possible data

Real-time insights
Billions of sensor nodes form a sensor cloud and provide data streams to analysis systems.
The Sensor Cloud – Problems

Billions of sensor nodes form a sensor cloud and provide data streams to analysis systems.

Sensor Nodes

Sensor Cloud

Stream Analysis Cluster

Real-time insights
The Sensor Cloud – Problems

Billions of sensor nodes form a sensor cloud and provide data streams to analysis systems.

Streaming all data from billions of sensors to all applications with maximal frequencies is impossible.
The Sensor Cloud – Problems

Billions of sensor nodes form a sensor cloud and provide data streams to analysis systems.

Streaming all data from billions of sensors to all applications with maximal frequencies is impossible.

Increasing data rates require expensive system scale-out.

Traub et al., Optimized On-Demand Data Streaming from Sensor Nodes, SoCC ‘17
The Sensor Cloud – Solutions

Tailor Data Streams to the Demand of Applications
The Sensor Cloud – Solutions

**Tailor Data Streams to the Demand of Applications**

- Provide an abstraction to define the data demand of applications.

  User-Defined Sampling Functions (UDSFs)
The Sensor Cloud – Solutions

**Tailor Data Streams to the Demand of Applications**

- Provide an abstraction to define the data demand of applications.
  - User-Defined Sampling Functions (UDSFs)
- Optimize communication costs while maintaining the result accuracy.
  - Read-Time Optimization
The Sensor Cloud – Solutions

**Tailor Data Streams to the Demand of Applications**

- Provide an abstraction to define the data demand of applications.
  
  - User-Defined Sampling Functions (UDSFs)

- Optimize communication costs while maintaining the result accuracy.
  
  - Read-Time Optimization

- Share sensor reads and data transfer among users and queries.
  
  - Multi-Query / Multi-User Optimization
A Motivating Example

Sensor Nodes

Data Center

speed, position, rpm
A Motivating Example

Sensor Nodes

- speed
- position
- rpm

Data Center

- Q1 demand

Query 1: Driver Profile
A Motivating Example

Different Data Demands:
• Query 1 adaptively increases sampling rates when accelerating or braking.
A Motivating Example

Different Data Demands:
- Query 1 adaptively increases sampling rates when accelerating or braking.
A Motivating Example

Different Data Data Demands:
- Query 1 adaptively increases sampling rates when accelerating or braking.
- Query 2 requires a sample at least every 20 meters
A Motivating Example

**Different Data Data Demands:**
- Query 1 adaptively increases sampling rates when accelerating or braking.
- Query 2 requires a sample at least every 20 meters
A Motivating Example

Different Data Data Demands:
• Query 1 adaptively increases sampling rates when accelerating or braking.
• Query 2 requires a sample at least every 20 meters
• Query 3 requires a sample at least every 0.3s.
A Motivating Example

Different Data Demands:
• Query 1 adaptively increases sampling rates when accelerating or braking.
• Query 2 requires a sample at least every 20 meters
• Query 3 requires a sample at least every 0.3s.
A Motivating Example

Different Data Data Demands:
• Query 1 adaptively increases sampling rates when accelerating or braking.
• Query 2 requires a sample at least every 20 meters
• Query 3 requires a sample at least every 0.3s.
A Motivating Example - Evaluation

(a) Number of sensor reads.  (b) Transferred tuples.
A Motivating Example - Evaluation

(a) Number of sensor reads.

(b) Transferred tuples.

-57%
A Motivating Example - Evaluation

(a) Number of sensor reads. (b) Transferred tuples.
A Motivating Example - Evaluation

(a) Number of sensor reads. (b) Transferred tuples.

1/3 because 3 values per tuple

-57%  
-72%
Architecture Overview

Sensor Nodes

A/D Sensors conv. Read Scheduler Operator push-down

\( S_1 \quad S_2 \ldots \quad S_n \)

User-Defined Sampling Functions (UDSFs)

Queries

Stream Analysis Cluster

per query

per sensor

Traub et al., Optimized On-Demand Data Streaming from Sensor Nodes, SoCC ‘17
Architecture Overview

Sensor Nodes

A/D Sensors
S_1
S_2
... S_n
conv.

Read
Scheduler

Operator
push-down

User-Defined Sampling Functions (UDSFs)

3 Join

2

PULL

PUSH

Stream
Analysis
Cluster

Queries

per query

per sensor

Traub et al., Optimized On-Demand Data Streaming from Sensor Nodes, SoCC ‘17
Architecture Overview

Sensor Nodes

A/D Sensors conv.

Read Scheduler

Operator push-down

Stream Analysis Cluster

User-Defined Sampling Functions (UDSFs)

Queries

S_1

S_2

S_n

PULL

PUSH

per query

per sensor

Traub et al., Optimized On-Demand Data Streaming from Sensor Nodes, SoCC '17
Architecture Overview

Sensor Nodes

A/D Sensors conv.

Read Scheduler

Operator push-down

User-Defined Sampling Functions (UDSFs)

Queries

Stream Analysis Cluster

per query

per sensor
Architecture Overview

Sensor Nodes

A/D Sensors conv. Read Scheduler Operator push-down

$S_1$, $S_2$, $S_n$

User-Defined Sampling Functions (UDSFs)

Queries

Stream Analysis Cluster

per query

per sensor

Traub et al., Optimized On-Demand Data Streaming from Sensor Nodes, SoCC ’17
Sensor Read Scheduling

Read Scheduler

UDSFs

read time suggestion
User-Defined Sampling Functions

**Read Scheduler**

Input:
Sensor read time and value

Output:
Next Sensor Read Request

\[ s : \langle t, v \rangle \rightarrow \langle t_{min}, t_D, t_{max}, p(t) \rangle \]
User-Defined Sampling Functions

Read Scheduler

Input: Sensor read time and value

\( s : \langle t, v \rangle \rightarrow \langle t_{\text{min}}, t_{\text{D}}, t_{\text{max}}, p(t) \rangle \)

- \( t_{\text{min}} \) - desired read time (\( t_{\text{D}} \))
- \( t_{\text{max}} \)
- ahead limit
- delay limit
- penalty function (\( p(t) \))
User-Defined Sampling Functions

Enable adaptive sampling techniques to reduce data transmission

e.g., Adam [Trihinas ‘15], FAST [Fan ‘14], L-SIP [Gaura ’13]
User-Defined Sampling Functions - Examples

\[ s : \langle t, v \rangle \rightarrow \langle t_{\text{min}}, t_D, t_{\text{max}}, p(t) \rangle \]

1: \textbf{upon sensor read }\langle \text{time, value} \rangle \textbf{ do}
2: \quad t_D \leftarrow \text{AdaM(time, value)} // \text{get next read time}
3: \quad t_{\text{min}} \leftarrow \max(\text{time}, t_D - 0.2s) // \text{get ahead limit}
4: \quad t_{\text{max}} \leftarrow t_D + 0.2s // \text{get delay limit}
5: \quad p(t) \leftarrow \text{abs}(t - t_D) // \text{set penalty function}
6: \quad \textbf{return } \langle t_{\text{min}}, t_D, t_{\text{max}}, p(t) \rangle
7: \quad \textbf{end}

Example 1: AdaM with 0.2s read time tolerance.
User-Defined Sampling Functions - Examples

\[ s : \langle t, \nu \rangle \rightarrow \langle t_{\text{min}}, t_D, t_{\text{max}}, p(t) \rangle \]

\[ s_{20m} : \langle t, \nu \rangle \rightarrow \langle t + 1, t + \frac{20m}{\nu}, t + \frac{20m}{\nu}, 0 \rangle \]

Example 2: Sample at least every 20 driven meter.
User-Defined Sampling Functions - Examples

\[ s : \langle t, v \rangle \rightarrow \langle t_{min}, t_{D}, t_{max}, p(t) \rangle \]

\[ s_{20m} : \langle t, v \rangle \rightarrow \langle t + 1, t + \frac{20m}{v}, t + \frac{20m}{v}, 0 \rangle \]

Example 2: Sample at least every 20 driven meter.

\[ s_{0.3s} : \langle t, v \rangle \rightarrow \langle t + 1, t + 0.3s, t + 0.3s, 0 \rangle \]

Example 3: Read a value at least every 0.3s.
Sensor Read Fusion

**Read Scheduler**

1. **UDSFs**
   - read time suggestion

2. **tolerance intervals**

3. **read fusion**

sensor → A/D
Sensor Read Fusion

Read Scheduler

1) Minimize Sensor Reads and Data Transfer:

Latest possible read time
Read Time Optimization

2) Optimize Sensor Read Times:

- Minimize penalty while executing the minimum number of sensor reads only
- Challenge: assign read requests to sensor reads
Assigning Read Requests to Sensor Reads

<table>
<thead>
<tr>
<th>Assign to next Read</th>
<th>Postpone</th>
</tr>
</thead>
</table>

Assigning Read Requests to Sensor Reads

Assign to next Read

<table>
<thead>
<tr>
<th></th>
<th>Postpone</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) \( \max(B) < \min(A) \Rightarrow \text{postpone.} \)
Assigning Read Requests to Sensor Reads

Assign to next Read

Postpone

\[(a) \max(B) < \min(A) \Rightarrow \text{postpone.}\]

\[(b) \max(A) < \min(B) \Rightarrow \text{assign to A.}\]
Assigning Read Requests to Sensor Reads

**Assign to next Read**

\[(b) \max(A) < \min(B) \Rightarrow \text{assign to } A.\]

**Postpone**

\[(a) \max(B) < \min(A) \Rightarrow \text{postpone}.\]

\[(c) \text{ otherwise } \Rightarrow \text{postpone}.\]
Local Filtering

Read Scheduler

1. UDSFs
   - read time suggestion

2. tolerance intervals
   - read fusion
   - sensor read time

3. read execution
   - local filter

sensor
A/D
Local Filtering

- Enable adaptive filtering in combination with adaptive sampling
- Enable model-driven data acquisition
Local Filtering

Read Scheduler

1. **UDSFs**
   - read time suggestion

2. tolerance intervals

3. sensor read time

4. read execution local filter

```
1: upon sensor read \langle time (t), value (v) \rangle do
2: \quad mv \leftarrow model.estimateValue(t)
3: \quad if abs(mv - v) > tolerance then
4: \quad \quad model.update(t, v) // local model update
5: \quad \quad return true // transfer value
6: \quad else
7: \quad \quad return false // no transfer required
8: \quad end if
9: end
```

Example 4: Local filter for model-driven data acquisition.
Evaluation

● Replay sensor data
  - from a football match [DEBS Grand Challenge ’13]
  - formula 1 telemetry data

● Random UDSFs:
  - Read in a poisson process (also simulate load peaks)
  - In average 1 read per query per second
  - Exponentially distributed read time tolerance
    - high probability for small tolerances
    - small probability for large tolerances
  - In average 0.04s read time tolerance
Increasing the number of concurrent queries

- On-Demand scheduling reduces sensor reads and data transfer by up to 87%.
- The # of reads and transfers increases sub-linearly with the # of queries.

(a) Number of sensor reads and data transmissions.
Increasing the number of concurrent queries

(b) Impact of read time optimization on read time deviations.

- Our read-time optimizer reduces the deviation from desired read times by up to 69% (preserving the min. # of reads and transfers).
Increasing read time tolerances

(a) Sensor reads/transfers.
Increasing read time tolerances

(a) Sensor reads/transfers.  
(b) Read time optimization.
Query Prioritization (1/2)

![Bar chart showing read time deviation for prioritized queries at different fractions](image)

**Figure 12:** Query prioritization with penalty functions. (20 queries; Øsampling rate 1Hz/query; Øtolerance ±0.04s)
Query Prioritization (2/2)

Figure 13: Read time optimization on behalf of a single query. Introductory use-case.
Slack Robustness of Adaptive Sampling

(a) Reads & Transfers
(b) Value Deviation

Figure 14: AdaM and FAST on football data with varying read time slack.
Wrap-Up:

Tailor Data Streams to the Demand of Applications

- Define data demand: User-Defined Sampling Functions
- Schedule sensor reads and data transfer on-demand
- Optimize read times globally - for all users and queries