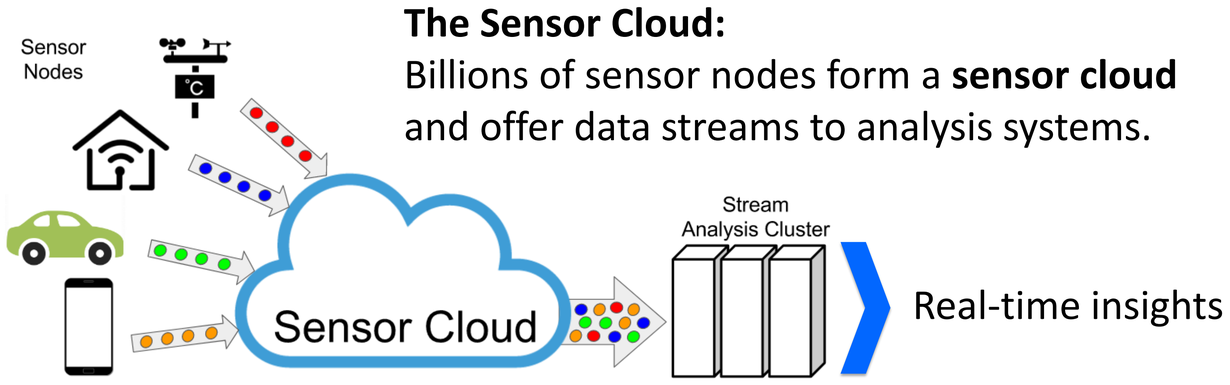
ACM Symposium on Cloud Computing, September 25-27, 2017, Santa Clara (CA)

Optimized On-Demand Data Streaming from Sensor Nodes

Jonas Traub jonas.traub@tu-berlin.de
Sebastian Breß sebastian.bress@dfki.de
Tilmann Rabl rabl@tu-berlin.de
Asterios Katsifodimos asterios.katsifodimos@sap.com
Volker Markl volker.markl@tu-berlin.de

Abstract



The Sensor Cloud:
Billions of sensor nodes form a **sensor cloud** and offer data streams to analysis systems.

Problem:

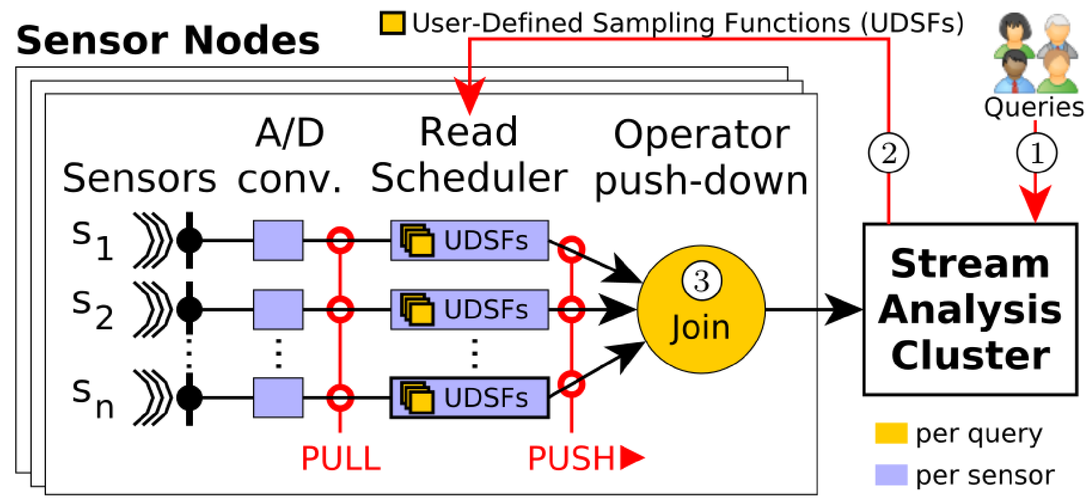
- Increasing data rates require expensive system scale-out.
- It is impossible to transfer all data from billions of sensors to all applications with maximal frequencies.

Solution:

Tailor Data Stream to the Demand of Applications

- Optimize communication costs while maintaining the result accuracy.
- Provide an abstraction to define the data demand of applications.
- Share sensor reads and data transfer among users and queries.

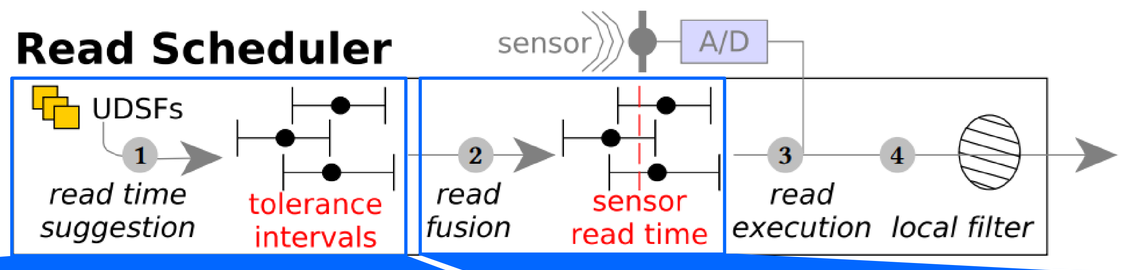
System Architecture



1. Users submit queries to a stream analysis cluster.
   - *We introduce user-defined sampling functions (UDSFs) to express the data-demand of queries.*
2. We propagate the UDSFs to the sensor nodes.
   - *Our Read Scheduler optimizes sensor read times and minimizes the data transfer based on the UDSFs.*
3. Push data to succeeding stream processing pipelines.

Multi-Query Read Scheduling

1. UDSFs propose read times with tolerance intervals.
2. We fuse proposed read times to a single sensor read if the tolerance intervals overlap.



3. We perform the actual read on the sensor.
4. We apply local filtering to further reduce data transmissions.

User-Defined Sampling Functions

An abstraction for the precise definition of each query's data demand

- Facilitate adaptive sampling techniques to reduce data transmission (e.g., Adam [Trihinas '15], FAST [Fan '14], L-SIP [Gaura '13]).
- Enable model-driven data acquisition [Deshpande '04, Raza '12].
- Diverse other application specific sampling strategies.

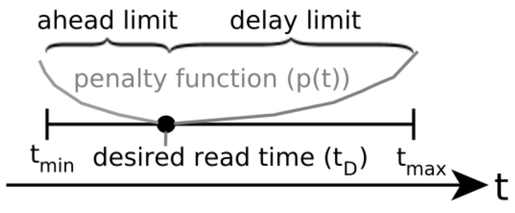
Syntax

- Upon a sensor read, propose next sensor read time.

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

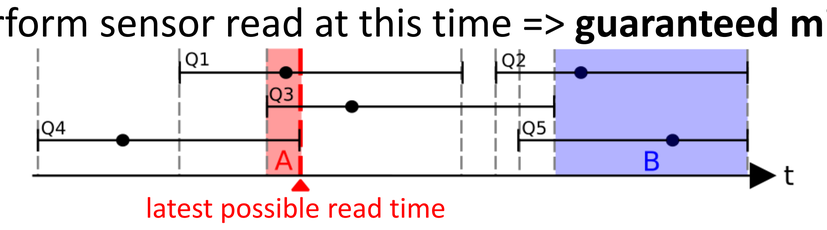
Input: Read time and sensor value

Output: Next sensor read time with tolerance interval and penalty function for read time deviations (used for optimization)

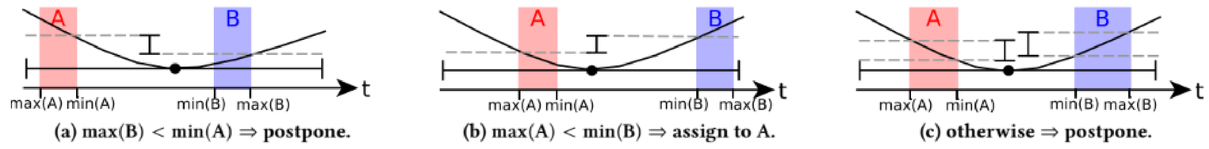


Read-Time Optimization

1. **Fuse proposed read times if tolerance intervals overlap.**
   - Find latest possible read time (first interval end).
   - Perform sensor read at this time => **guaranteed min. # of reads.**

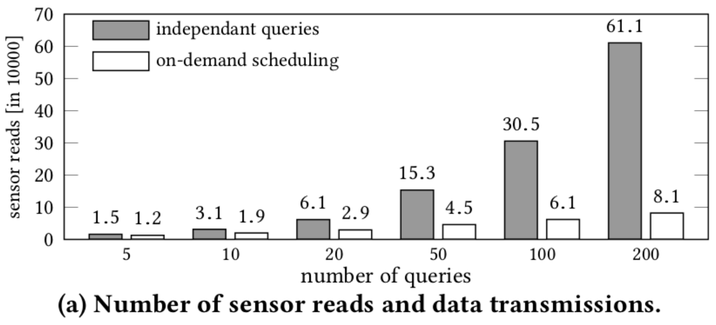


2. **Optimize read times while preserving the min. # of reads in total.**
   - Determine time intervals in which we will perform sensor reads.
   - Assign requested sensor reads to these intervals (A or B).
   - Minimize penalty to find the best sensor read time.

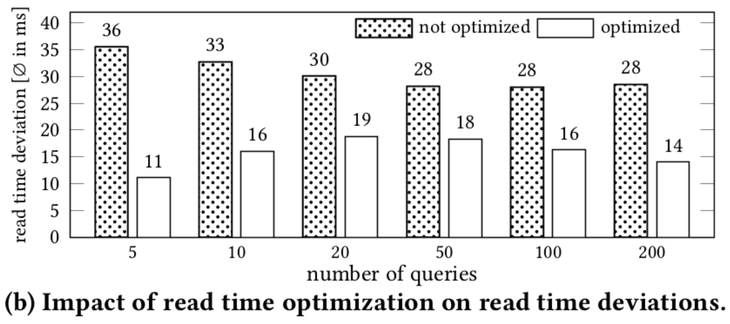


(a) max(B) < min(A) ⇒ postpone. (b) max(A) < min(B) ⇒ assign to A. (c) otherwise ⇒ postpone.

Evaluation



(a) Number of sensor reads and data transmissions.



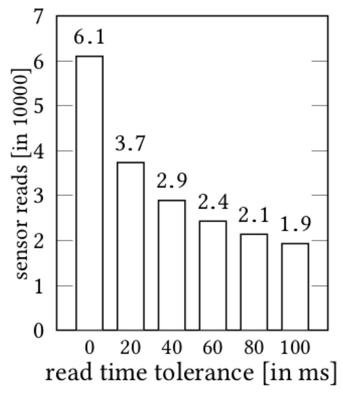
(b) Impact of read time optimization on read time deviations.

Scaling to large numbers of concurrent queries

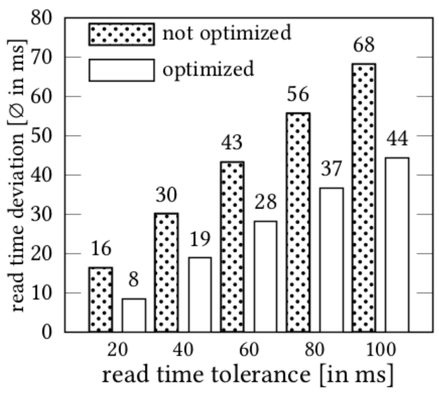
- On-Demand scheduling reduces sensor reads and data transfer by up to 87%.
- The number of reads and transfers increases sub-linearly when raising the number of queries.
- Our read-time optimizer reduces the deviation from desired read times by up to 69% (preserving the min. # of reads and transfers).

Increasing Read-Time Tolerances

- Even small tolerances enable huge savings.
- Read-time optimization is always beneficial.



(a) Sensor reads/transfers.



(b) Read time optimization.

Technische Universität Berlin
German Research Center for Artificial Intelligence

This work received funding through the EU Horizon 2020 projects Streamline (688191) and SAGE (671500) and from the German Ministry for Education and Research as Berlin Big Data Center (01IS14013A) and Software Campus (01IS12056).