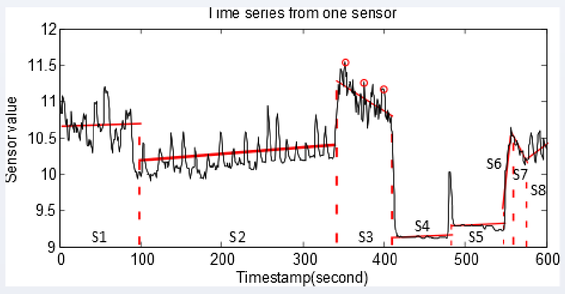
Online indexing and distributed querying model-view sensor data in the cloud

Tian Guo[1], Thanasis G. Papaioannou[2], Hao Zhuang[1], Karl Aberer[1]

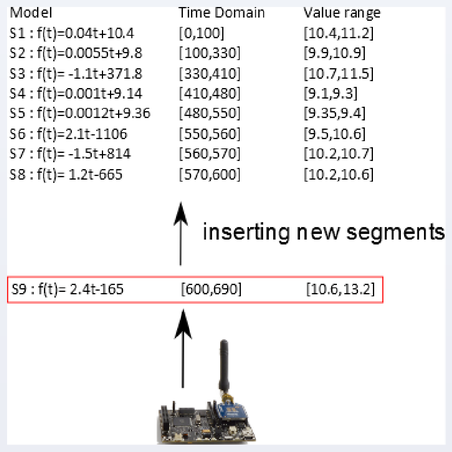
1. EPFL (Swiss Federal Institute of Technology, Lausanne)
2. Center for Research & Technology Hellas (ITI-CERTH)

Model-view sensor data in the cloud

Sensor time series segmentation



Each segment approximated by one model



Queries of interest:

Time(or value) range queries: SELECT sensor values (or time ranges) FROM sensor 1 WHEN t1 <= time <= t2 (or v1 <= value <= v2)

Time(or value) point queries: SELECT sensor values (or time ranges) FROM sensor 1 WHEN time = t (or value = v)

Composite queries: SELECT sensor values (or time ranges) FROM sensor 1 WHEN t1 <= time <= t2 AND v1 <= value <= v2

Challenges for managing model-view sensor data in the cloud

Store model-view sensor data in the distributed cloud store

Challenges:

Storage scheme

- ❑ Model-view sensor data in key-value stores.

Model index

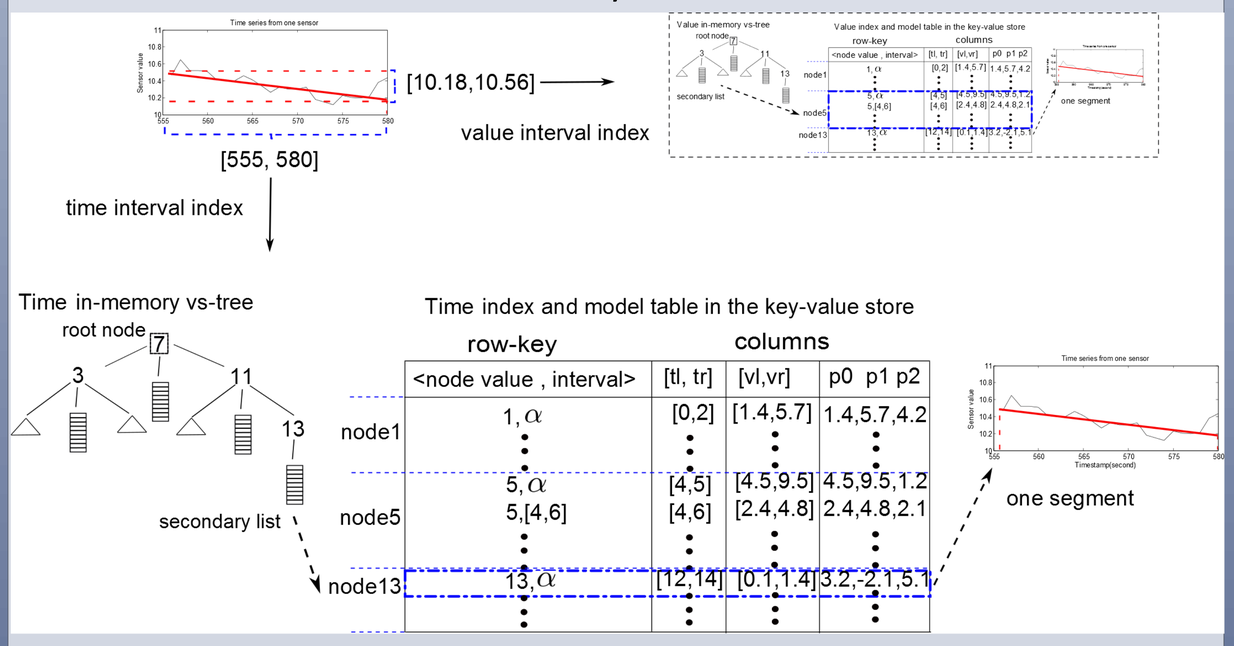
- ❑Update the segments of model-view sensor data on the fly
- ❑Support various queries over model-view sensor data via distributed computing

Key-Value model index

Index structure:

(1) Index time and value dimension respectively.

(2) two-tire structure of time (or value) index:
in-memory virtual search tree (vs-tree)
index and model table in the key-value store



Registration node of one segment:

(1) It is the highest one that overlaps the interval to index on vs-tree.

(2) It is the node of which secondary structure one segment is materialized into.

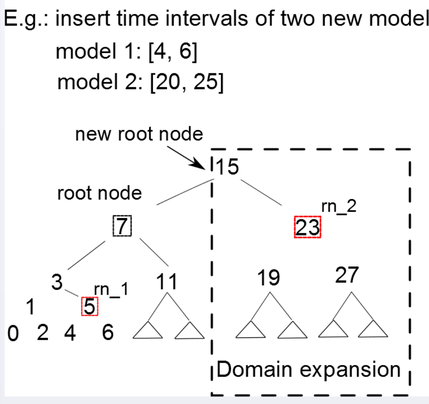
Index update:

(1) in-memory search on vs-tree.

(2) segment materialization in the key-value store.

Registration node searching on vs-tree

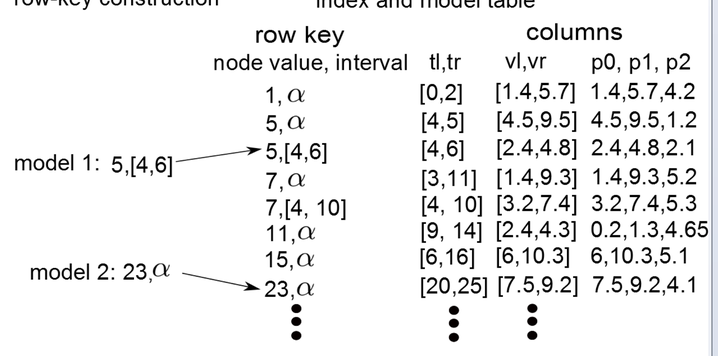
E.g.: insert time intervals of two new models
model 1: [4, 6]
model 2: [20, 25]



Segment materialization

row-key construction

index and model table

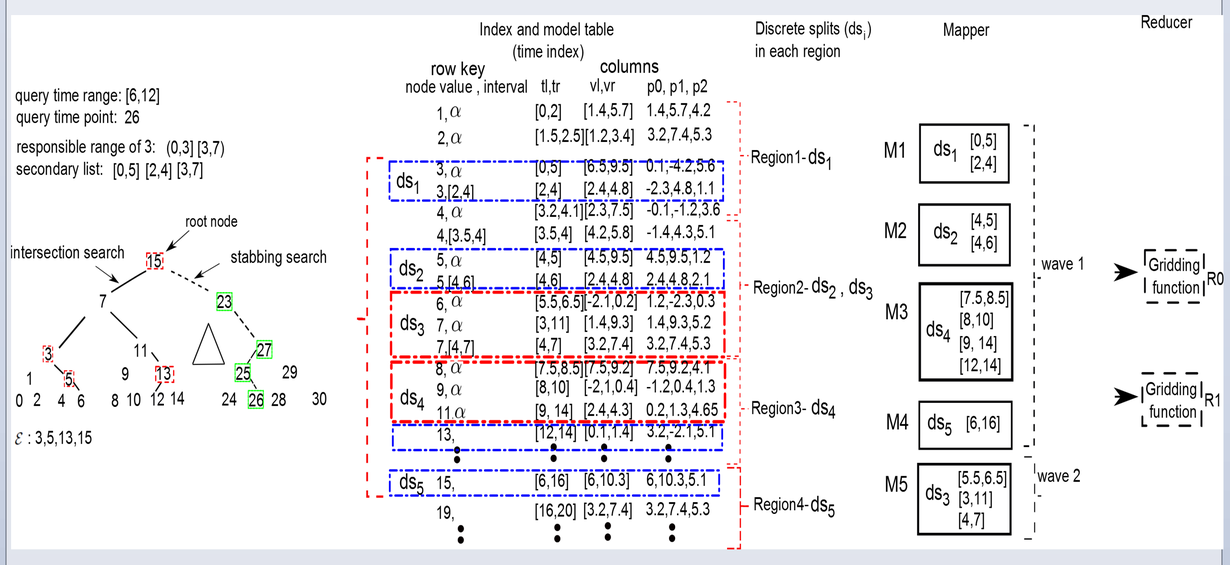


Query processing

Point and Range queries:

(1) search on vs-tree to find relevant nodes.

(2) modify the data access method of MapReduce to enable to process the discrete parts of the index and model table.



Composite queries: adaptive index selection decides which index-model table to be processed by MapReduce.

Based on the splits n_s given the maximum mappers slots M waves is defined as

$$W = \frac{n_s}{M}$$

data transferring cost is defined as

$$D_t = \sum_{i=1}^{n_r} g(\lfloor W \rfloor * m_i - |S_i|) \qquad g(x) = \begin{cases} 0, x >= 0 \\ |x|, otherwise \end{cases}$$

the number of discrete splits $|S_i|$ in each region

n_r is the number of regions of the index-model table.

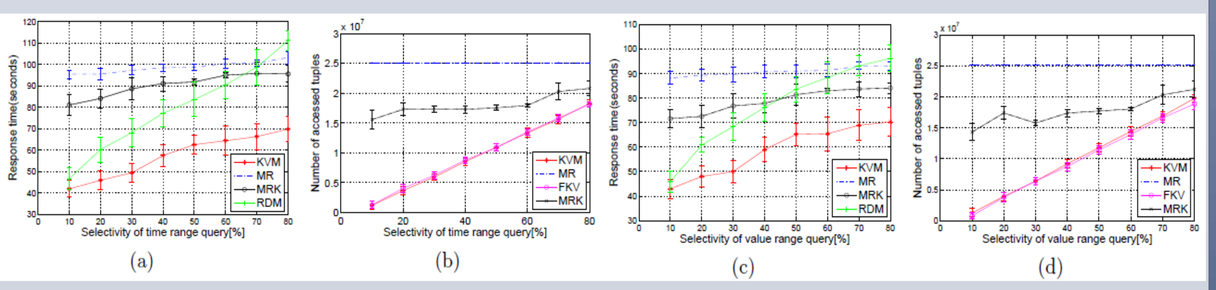
mapper slots m_i of each node

the combined cost of one composite query for time (or value) index is defined as

$$C = \alpha * W + (1 - \alpha) D_t, (0 \leq \alpha \leq 1)$$

Experimental evaluation

Comparison of range query performance:



Acknowledgements

This research is supported in part by the EU OpenIoT Project FP7-ICT-2011-7-287305 and OpenSense project under SNF Number 20NA21 128839