Big Data Research 1 (2014) 52–65

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Efficient Indexing and Query Processing of Model-View Sensor Data in the Cloud ☆

Tian Guo [a,*], Thanasis G. Papaioannou [b], Karl Aberer [a]

[a] EPFL, Switzerland
[b] Center for Research & Technology Hellas (CERTH), Greece

ARTICLE INFO

Article history:
Available online 11 July 2014

Keywords:
Big data
Index
Key-value stores
MapReduce
Approximation
Query optimization

ABSTRACT

As the number of sensors that pervade our lives increases (e.g., environmental sensors, phone sensors, etc.), the efficient management of massive amount of sensor data is becoming increasingly important. The infinite nature of sensor data poses a serious challenge for query processing even in a cloud infrastructure. Traditional raw sensor data management systems based on relational databases lack scalability to accommodate large-scale sensor data efficiently. Thus, distributed key-value stores in the cloud are becoming a prime tool to manage sensor data. Model-view sensor data management, which stores the sensor data in the form of modeled segments, brings the additional advantages of data compression and value interpolation. However, currently there are no techniques for indexing and/or query optimization of the model-view sensor data in the cloud; full table scan is needed for query processing in the worst case. In this paper, we propose an innovative index for modeled segments in key-value stores, namely KVI-index. KVI-index consists of two interval indices on the time and sensor value dimensions respectively, each of which has an in-memory search tree and a secondary list materialized in the key-value store. Then, we introduce a KVI-index–Scan–MapReduce hybrid approach to perform efficient query processing upon modeled data streams. As proved by a series of experiments at a private cloud infrastructure, our approach outperforms in query-response time and index-updating efficiency both Hadoop-based parallel processing of the raw sensor data and multiple alternative indexing approaches of model-view data.

1. Introduction

Recent advances in sensor technology have enabled the vast deployment of sensors embedded in user devices that monitor various phenomena for different applications of interest, e.g., air/electrosmog pollution, radiation, early earthquake detection, soil moisture, permafrost melting, etc. The data streams generated by a large number of sensors are represented as time series in which each data point is associated with a time-stamp and a sensor value. These raw discrete observations are taken as the first citizen in traditional relational sensor data management systems, which leads to a number of problems. On one hand, in order to perform analysis of the raw sensor data, users usually adopt other third-party modeling tools (e.g., Matlab, R and Mathematica), which involve of tedious and time-consuming data extract, transform and load (ETL) processes [1]. Moreover, such data analysis tools are usually used for static data set and therefore cannot be applied for online processing of sensor data streams. On the other hand, unbounded sensor data streams often have missing values and unknown errors, which also poses great challenges for traditional raw sensor data management.

To this end, various model-based sensor data management techniques [1–4] have been proposed. Model-view sensor data management leverage time series approximation and modeling techniques to segment the continuous sensor time series into disjoint intervals such that each interval can be approximated by a kind of model, such as polynomial, linear or SVD. These models, for all the intervals (or segment models), exploit the inherent correlations (e.g. with time or among data streams) in the segments of sensor time series and approximate each segment by a certain mathematical function within a certain error bound [5–9]. Then, one can only materialize the models of the segments instead of the raw data and harvest a number of benefit:

First, model-view sensor data achieves compression over raw sensor data and therefore requires less storage overhead [10–12]. Second, due to the sampling frequency or sensor malfunction,

☆ This article belongs to Scalable Computing for Big Data.

* Corresponding author.
E-mail addresses: tian.guo@epfl.ch (T. Guo), thanasis.papaioannou@iti.gr (T.G. Papaioannou), karl.aberer@epfl.ch (K. Aberer).

http://dx.doi.org/10.1016/j.bdr.2014.07.005
2214-5796/

there may be some missing values at some time points. If one query involves such time points, then the relevant segment model can be used to estimate the values [10–12]. In some degree, model-view sensor data increases the data availability for query processing. Third, there usually exist outliers in raw sensor data, which has negative effect on the query results. Model-view sensor data removes the outliers in each interval via the segment model and historical data on upper and lower data bounds, thereby diminishing the effect of outliers in query results. Fourth, regarding the similarity search or pattern discovery in sensor time-series mining, the segment-based time series representation is a powerful tool for dimension reduction and search space pruning [9,13,14].

However, proposed model-based sensor data management approaches mostly employ the relational data model and process queries based on materialized views [2,3] on top of the modeled segments of sensor data. Nowadays, the amount of data produced by sensors is exponentially increasing. Moreover, the real-time production of sensor data requires the data management system to be able to handle high-concurrent model-view sensor data from massive sensors and this is difficult for traditional relational database to realize. To this end, recent prevalent cloud store and computing techniques provide a promising way to manage model-view sensor data [15–19].

The main focus of this paper is on how to manage the segment models of sensor data, namely model-view sensor data with the newly emerging cloud stores and computing techniques rather than how to explore more advanced sensor data segmentation algorithms.

In our approach, we exploit key-value stores and the MapReduce parallel computing paradigm [18,20], two significant aspects of cloud computing, to realize indexing and querying model-view sensor data in the cloud. We characterize the modeled segments of sensor time series by the time and the value intervals of each segment [16,17]. Consequently, in order to process range or point queries on model-view sensor data, our index in the cloud store should excel in processing interval data. Current key-value built-in indices do not support interval-related operations. The interval index for model-view sensor data should not only work for static data, but it should be dynamically updated based on the new arriving segments of sensor data. If traditional batch-updating or periodical re-building strategy was applied here [21,22], then the high speed of sensor data yielding might lead to a large size of the new unindexed data, even in short time periods and to significant index updating delay as well. As a result, the performance of queries involving both indexed and unindexed data would degenerate greatly. Therefore, the interval index in the cloud store should be able to insert an individual new modeled segment in an online manner.

The contributions of this paper are summarized as follows:

- Innovative interval index: We propose an innovative interval index for model-view sensor data management in key-value stores, referred to as *KVI-index*. *KVI-index* is a two-tier structure consisting of one lightweight and memory-resident binary search tree and one index-model table materialized in the key-value store. This composite index structure can dynamically accommodate new sensor data segments very efficiently.
- Hybrid model-view query processing: After exploring the search operations in the in-memory structure of the KVI-index for range and point queries, we introduce a hybrid query processing approach that integrates range scan and MapReduce to process the segments in parallel.
- Intersection search: We introduce an enhanced intersection search algorithm (*iSearch+*) that produces consecutive results suitable for MapReduce processing. We theoretically analyze the efficiency of (*iSearch+*) and find the bound on the redundant index nodes that it returns.
- Experimental evaluation: Our framework has been fully implemented, including the online sensor data segmentation, modeling, KVI-index and the hybrid query processing, and it has been thoroughly evaluated against a significant number of alternative approaches. As experimentally shown based on real sensor data, our approach significantly outperforms in terms of query response time and index updating efficiency all other ones for answering time/value point and range queries.

The remainder of this paper is organized as follows: Section 2 summarizes some related work on model-view sensor data management, interval index and index-based MapReduce optimization approaches. In Section 3, we provide a brief description of sensor-data segmentation, querying model-view sensor data and the necessity to develop interval index for managing model-view sensor data in key-value stores. The detailed designs of our innovative KVI-index and the hybrid query processing approach are discussed in Sections 4 and 5 respectively. Then, in Section 6, we present thorough experimental results to evaluate our approach with traditional query processing ones on both raw sensor data and modeled data segments. Finally, in Section 7, we conclude our work.

2. Related work

Time series segmentation is an important research problem in the areas of data approximation, data indexing and data mining. A lot of work has been devoted to exploit different types of models to approximate the segments of time series, such that the pruning and refinement framework can be applied to this segment-represented time series for the pattern matching or similarity search [9,13]. Some other researchers proposed techniques for managing the segment models that approximate sensor time series in relational databases. MauveDB designed a model-based view to abstract underlying raw sensor data; it then used models to project the raw sensor readings onto grids for query processing [3]. As opposed to MauveDB, FunctionDB only materializes segment models of raw sensor data [2]. Symbolic operators are designed to process queries using models rather than raw sensor data. However, both approaches in [3] and [2] do not take into account applying indices to improve query processing. Moreover, their proposed approach focuses on managing static dataset of time series rather than dynamic time series.

Also, each segment of time series could be characterized by its time and value intervals. Then, one should consider employing an interval index for processing queries over model-view sensor data. Two common used indices for interval data are segment tree [21] and interval tree [23]. As for segment tree, it is essentially a static data structure that needs an initialization process to construct elementary intervals based on the initial dataset [21]. Once a new interval outside of the domain of current segment tree arrives, the elementary intervals should be rebuilt, which is not suitable for the real time nature of sensor data streams [21]. Regarding the interval tree, individual interval is not divided and replicated during the insertion phase as in the segment tree and therefore the storage overhead is linear to the number of intervals to index [23–25]. However, it is a memory-oriented structure.

Some efforts [22,23,26,27] have also been done to externalize these in-memory index data structures. The relational interval tree (RI-tree) [26] integrates interval tree into relational tables and transforms interval queries into SQL queries on tables. This method makes efficient use of built-in B+-tree indices of RDBMS. Nevertheless, in this paper, we aim to design an interval index structure for model-view sensor data that is compatible with key-value stores and distributed query processing in the cloud.