Decaying Telco Big Data with Data Postdiction

Constantinos Costa*, Andreas Charalampous*, Andreas Konstantinidis*‡,
Demetrios Zeinalipour-Yazti* and Mohamed F. Mokbel§

*Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus
‡Department of Computer Science & Engineering, Frederick University, 1036 Nicosia, Cyprus
§Qatar Computing Research Institute, HBKU, Qatar and University of Minnesota, Minneapolis, MN 55455, USA
{costa.c, achara28, akonstan, dzeina}@cs.ucy.ac.cy; mmokbel@hbku.edu.qa

***Abstract*—In this paper, we present a novel decaying operator for Telco Big Data (TBD), coined *TBD-DP (Data Postdiction)*. Unlike data prediction, which aims to make a statement about the future value of some tuple, our formulated *data postdiction* term, aims to make a statement about the past value of some tuple, which does not exist anymore as it had to be deleted to free up disk space. *TBD-DP* relies on existing Machine Learning (ML) algorithms to abstract TBD into compact models that can be stored and queried when necessary. Our proposed *TBD-DP* operator has the following two conceptual phases: (i) in an offline phase, it utilizes a LSTM-based hierarchical ML algorithm to learn a tree of models (coined *TBD-DP* tree) over time and space; (ii) in an online phase, it uses the *TBD-DP* tree to recover data within a certain accuracy. In our experimental setup, we measure the efficiency of the proposed operator using a ∼10GB anonymized real telco network trace and our experimental results in Tensorflow over HDFS are extremely encouraging as they show that *TBD-DP* saves an order of magnitude storage space while maintaining a high accuracy on the recovered data.**

***Index Terms*—telco, big data, spatio-temporal analytics, data decaying, data reduction, machine learning.**

I. Introduction

In recent years there has been considerable interest from *telecommunication companies (telcos)* to extract concealed value from their network data. Consider for example a telco in the city of Shenzhen, China, which serves 10 million users. Such a telco is shown to produce 5TB per day [1] (i.e., thousands to millions of records every second). Huang et al. [2] break their 2.26TB per day *Telco Big Data (TBD)* down as follows: (i) *Business Supporting Systems (BSS)* data, which is generated by the internal work-flows of a telco (e.g., billing, support), accounting to a moderate of 24GB per day and; (ii) *Operation Supporting Systems (OSS)* data, which is generated by the Radio and Core equipment of a telco, accounting to 2.2TB per day and occupying over 97% of the total volume.

Effectively storing and processing TBD workflows can unlock a wide spectrum of challenges, ranging from churn prediction of subscribers [2], city localization [3], 5G network optimization / user-experience assessment [4]–[6] and road traffic mapping [7]. Even though the acquisition of TBD is instrumental in the success of the above scenarios, Telcos are reaching a point where they are collecting more data than they could possibly exploit. This has the following two implications: (i) it introduces a significant financial burden on the operator to store the collected data locally. Notice that the deep storage of data in public clouds, where economies-of-scale are available (e.g., AWS Glacier), is not an option due to privacy reasons; and (ii) it imposes a high computational cost for accessing and processing the collected data. For example, a petabyte Hadoop cluster, using between 125 and 250 nodes, costs ∼1M USD [8] and a linear scan of 1PB would require almost 15 hours. Additionally, in [9] it is shown that the amount of storage doubles every year and storage media costs decline only at a rate of less than 1/5 per year. Finally, high-availability storage mandates low-level data replication (e.g., in HDFS the default data replication is 3).

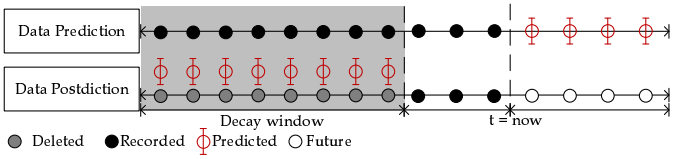

Fig. 1. **Data Prediction (top)**: aims to find the future value of some tuple. **Data Postdiction (bottom)**: aims to recover the past value of some tuple, which has been deleted to reduce the storage requirements, using a ML model.

Consequently, we claim that the vision of infinitely storing all IoT-generated velocity data on fast high-availability or even deep storage will gradually become too costly and impractical for many analytic-oriented processing scenarios.

To this end, *data decaying* [10], [11] (or data rotting) has recently been suggested as a powerful concept to complement traditional data reduction techniques [12], [13], e.g., sampling, aggregation (OLAP), dimensionality reduction (SVD, DFT), synopsis (sketches) and compression. Data decaying refers to *"the progressive loss of detail in information as data ages with time"*. In data decaying recent data retains complete resolution, which is practical for operational scenarios that can continue to operate at full data resolution, while older data is either compacted or discarded [5], [10], [11]. Additionally, the decaying cost can be amortized over time, matching current trends in micro-batching (e.g., Apache Spark). Unfortunately, data decaying currently relies on rather straightforward methodologies, such as rotational decaying (i.e., FIFO) [10], or decaying based on specific queries [5] rather than the complete dataset itself. Our aim in this work is to expand upon these developments to provide more intelligent and generalized decaying operators.

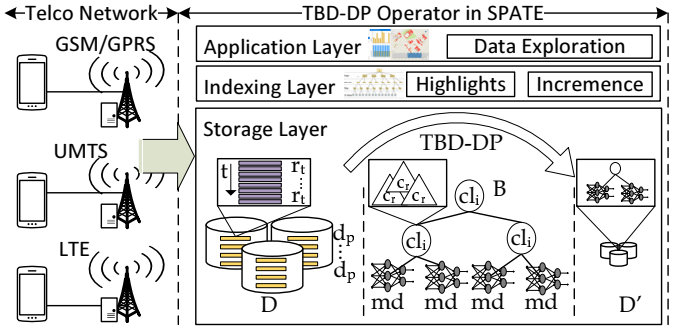

Fig. 2. System Model: The TBD-DP operator works on the storage layer of a typical TBD stack and abstracts the incoming data signals (D) into abstract models (md) that are organized in a tree data structure (B).

In this paper, we present a novel decaying operator for Telco Big Data, coined *TBD-DP (Data Postdiction)* (see Figure 1). Unlike data prediction, which aims to make a statement about the future value of some tuple in a TBD store, data postdiction aims to make a statement about the past value of some tuple that does not exist anymore, as it had to be deleted to free up space. *TBD-DP* relies on existing Machine Learning (ML) algorithms to abstract TBD into compact models that can be stored and queried when necessary. Our proposed *TBD-DP* operator has the following two conceptual phases: (i) in an offline phase, it utilizes a LSTM-based hierarchical ML algorithm to learn a tree of models (coined *TBD-DP* tree) over time and space; (ii) in an online phase, it uses the *TBD-DP* tree to recover data with a certain accuracy.

We claim that the LSTM model is capturing the essence of the past through its short and long-term dependencies, similarly to how the brain retains both recent information and important old information at a high resolution.

To understand the operational aspects of our proposed TBD-DP operator, consider Figure 2, where we show how incoming telco data signals are absorbed by the TBD architecture and stored on high-availability and fast storage (i.e., D). This helps to carry out operational tasks (e.g., alerting services and visual analytics) with full data resolution. Subsequently, in the first phase of *TBD-DP*, we utilize a specialized *Recurrent Neural Network (RNN)* composed of *Long Short Term Memory (LSTM)* units, which has the ability to detect long-term correlations in activity data and the trained model has a small disk space footprint [14]. This enables *TBD-DP* to utilize minimum storage capacity of the decayed data by representing them with LSTM models on the disk media (D') and provide real-time postdictions with high accuracy in a subsequent recovery phase, which will be initiated on-demand (i.e., whenever some high-level operator requests the given data blocks).

The contributions of this work are summarized as follows:

- We propose a TBD decay operator that deploys the notion of data postdiction using off-the-shelf LSTM-based prediction models.
- We propose the DP-tree, which is a hierarchical index to organize the generated models in a data structure to enable the efficient recovery of data when necessary.

TABLE I
SUMMARY OF NOTATION

Notation	Description
p, d_p, D	Ingestion period, data snapshot of one p, set of all d_ps
t, r_t	Timestamp within an ingestion cycle, record at t
C, c_r, cl_i	Set of all cell towers, Cell of record r, cluster of records $i = 1, \ldots, k$
md_i, MD	LSTM model of cluster cl_i, set of all models
f	Decaying factor: percentage of data to be removed

- We measure the efficiency of the proposed operator using a ∼10GB anonymized telco network trace, showing that *TBD-DP* can be a premise for efficient TBD analytics in the future. We also summarize a prototype architecture and user interface we have developed for the management of TBD.

II. System Model and Problem Formulation

This section formalizes our system model, assumptions and problem. The main symbols and their respective definitions are summarized in Table I.

A typical Telco system, illustrated in Figure 2, is composed of the Telco network, which is responsible for providing telecommunication services, and a Telco data management system, such as SPATE [5], which is responsible for the efficient analytical exploration of Telco datasets. The data arrives at the data center in batches, called henceforth data snapshots noted by d_p, in the form of horizontally segmented files within an ingestion period p. A snapshot d_p contains multiple records r_t created at a certain timestamp t. Each record r_t consists of a predefined set of attributes including the cell id c_r that represents the spatial information inherent within the Telco network. Particularly, each cell id c_r corresponds to a cell that covers a geographical cellular area that usually spans hundreds of meters or even kilometers. Finally, the cells are spatially grouped into clusters $cl_i, i = i \ldots k$ for facilitating the postdiction process by creating a model $md_i, i = i \ldots k$ for each cl_i as this will be explained in the next section.

A. Problem Formulation

Research Goal. *Given a Telco setting, this work aims at achieving a pre-specified decaying of TBD with minimum additional storage space capacity and being able to recover the decayed data accurately and efficiently.*

The efficiency of the proposed techniques to achieve the above goal is measured by the following objectives:

Definition 2.1: **Storage Capacity** (S) is the total storage space required for achieving decaying of data based on a pre-specified decaying factor f.

Definition 2.2: **Accuracy** (***NRMSE***) is the percentage of the correctly recovered decayed data. It is measured by the normalized root-mean-square error, which is the normalized difference between the actual data ($x_{1,t}$) and the predicted

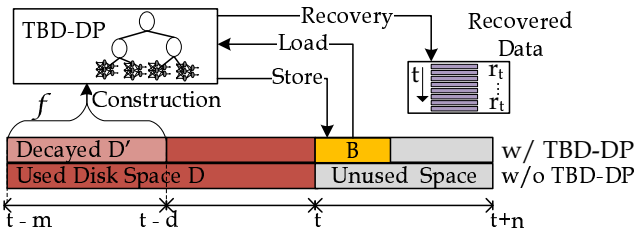

Fig. 3. *TBD-DP* Operator Overview.

data ($x_{2,t}$), where t is a discrete time point and y_{max}, y_{min} the maximum and minimum observed differences, formally:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(x_{1,t} - x_{2,t})^2}}{(y_{max} - y_{min})}$$

III. The *TBD-DP* Operator

In this section, we introduce the proposed *TBD-DP* operator and discuss its two internal algorithms, namely, the *Construction* (data model creation) and the *Recovery* (data recreation), which capture its core functionality as illustrated in Figure 3.

The *Construction* algorithm can be triggered either by the user, or automatically when the total storage capacity reaches a certain level. In both cases, the data are initially clustered based on spatial characteristics and then ordered based on temporal information. Finally, postdiction models based on the LSTM machine learning approach are generated for each cluster and the real data is decayed by $f\%$. The *Recovery* algorithm utilizes the postdiction models for retrieving the decayed data by adopting a proposed DP-tree based algorithm.

A. Construction Algorithm

Algorithm 1 outlines the major steps of the construction algorithm. Initially, the decaying factor f specifies the percentage of the whole dataset D that will be decayed, and consequently the decayed subset $D' \subseteq D$ that will be utilized for generating the postdiction models. In the spatial partitioning step (Step 1 - lines 2-5), $k \leq |C|$ clusters are created by using the cell tower locations. Particularly, each cluster $cl_i, i = 1, \ldots, k$ is represented by a cell tower (in cases where $k < |C|$ then the closest cell towers are merged using a kNN approach until we finally generate k clusters). This allows us to construct less models based on the network topology resulting to less computations. Then the MAP function associates all records $r_t \in D'$ with the previously created clusters by taking into consideration their cell id c_r attribute. By the end of this function execution, k clusters of cell towers with their associate records will be created. Then all records of each cluster are ordered based on their timestamp (i.e., time originally generated) by using the *ORDER* function of the temporal ordering step (Step 2 - lines 6-8). This allows the neural network to be created correctly based on a continuous time series. Finally, the learning step (Step 3 - lines 9-12) generates k postdiction models md_i for each cluster cl_i by using a specialized Recurrent Neural Network (RNN) known as the Long Short Term Memory (LSTM) model [15].

Algorithm 1 - *TBD-DP* Construction Algorithm

Input: Dataset D, C set of cell towers, Number of clusters k
Output: B: Set of models MD (DP-tree structure)

▷ **Step 0: Decaying Pre-processing**
1: $D' \leftarrow f$ of D ▷ Select $f\%$ of D to be decayed
▷ **Step 1: Spatial Partitioning**
2: Create $k \leq |C|$ clusters cl_i ▷ Use cell towers locations
3: **for all** $r_t \in D'$ **do**
4: $cl_i \leftarrow MAP(r_t, cl_i)|i = 1, \ldots k$ ▷ Associate records to clusters
5: **end for**
▷ **Step 2: Temporal Ordering**
6: **for** i = 1 to k **do**
7: $cl_i \leftarrow ORDER(cl_i)$ ▷ Sort records in clusters based on timestamp
8: **end for**
▷ **Step 3: Hierarchical Model**
9: **for** i = 1 to k **do**
10: $md_i \leftarrow LEARNING(cl_i)$ ▷ Create an LSTM model for each cl_i
11: Insert md_i in B
12: **end for**

Specifically, the *LEARNING* function generates, for each cluster at each iteration, an LSTM model that relies on a structure called a memory cell, which is composed of four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. A memory cell is updated at every time-step by using the following parameters and equations:

- x_t is the input to the memory cell layer at time-step t
- W_i, W_f, W_C and W_o are weight matrices
- b_i, b_f, b_C and b_o are bias vectors

The forget gate layer:

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f),$$

decides what information are going to be thrown away from the memory cells. The input gate layer:

$$i_t = s(W_i[h_{t-1}, x_t] + b_i),$$

decides which values to be updated. The tanh layer decides what new information we are going to store in the memory cells using:

$$\widetilde{C_t} = tanh(W_C[h_{t-1}, x_t] + b_C).$$

Moreover, the update memory cells function:

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C_t},$$

used to forget the things decided to be forgotten earlier and scale the new candidate values by a pre-specified state value.

Finally, the update hidden cells function:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

and a sigmoid layer that decide what parts of the cell state to output,

$$h_t = o_t * tanh(C_t).$$

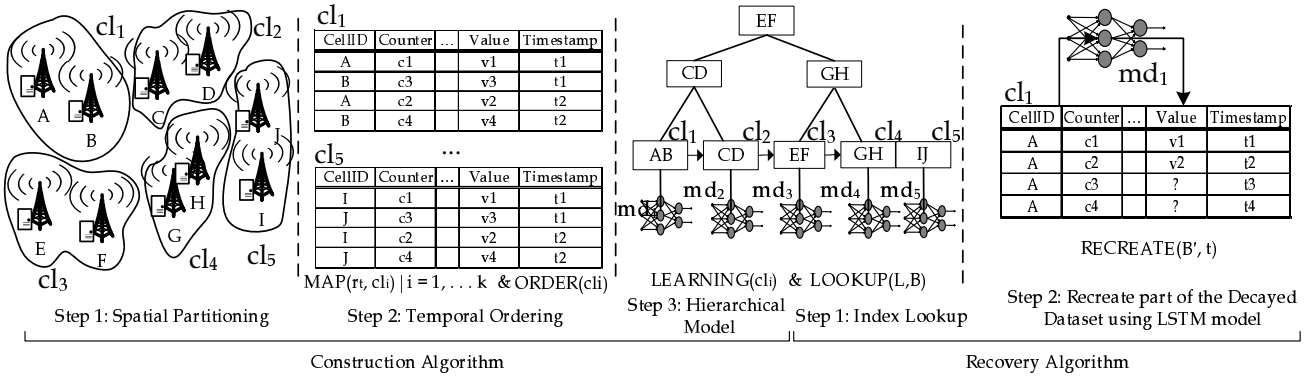

Fig. 4. The conceptual steps of the proposed *TBD-DP* construction and recovery algorithm.

The *Construction* algorithm outputs a set of postdiction models B in a DP-tree for facilitating the recovery algorithm that follows. At the end of the *Construction* algorithm execution, the D' set of data is removed for saving storage space and it is conceptually replaced by the final B set of postdiction models, where $|B| << |D'|$.

Example: Consider the scenario in Figure 4 where there are 10 cell towers $\{A, ..., J\}$. First, the *Construction* algorithm creates $k = 5$ clusters $\{cl_1,...,cl_5\}$ denoted with the solid line that surrounds the cell towers in Step 1 of Figure 4 (left). The *MAP* function associates the records to a cluster based on the cell id c_r (e.g., all records related to *A* and *B* are grouped into cl_1). Then, the *ORDER* function sorts the records of each cluster based on their timestamp t as shown in Step 2 of Figure 4 (center). Finally, for each cluster cl_i a model md_i is trained and inserted into a DP-tree index using the cell ids as keys, as shown in Figure 4 (right).

Decay Principle of TBD-DP: *Decaying* refers to the progressive loss of detail in information as data ages with time until it has completely disappeared. Kersten refers to the existence of data fungus in [11] with a decaying operator coined *"Evict Grouped Individuals (EGI)"*. The given EGI operator performs biased random decaying, resembling the rotting process in nature (e.g., in fruits with fungus). In our previous work [5], we used the *First-In-First-Out (FIFO)* data fungus, i.e., *"Evict Oldest Individuals"*, which retains full resolution for recent data but abstracts older data into compact aggregation models. Both EGI and FIFO do not retain full resolution for important instances that occurred in the past. Consequently, data would have been rotted and purged either randomly or based on its timestamp. We call this the *long-term dependency* problem. In this work, we chose a radically new decaying technique that could be termed as *LSTM data fungus*, which is explicitly designed to avoid the *long-term dependency* problem. Particularly, the *TBD-DP* operator replaces the data with abstract LSTM models, which capture the essence of the past, i.e., both recent data and important old data is retained at the highest possible resolution.

B. Recovery Algorithm

Algorithm 2 outlines the *Recovery* algorithm that utilizes the DP-tree structure (B) of postdiction models of Algorithm 1

Algorithm 2 - *TBD-DP* Recovery Algorithm

Input: L: spatial input; R: temporal input; B: set of postdiction models in a DP-tree structure
Output: Partial decayed dataset pD'

1: **procedure** LOOKUP(k,$node$) ▷ The number of children is b.
2: **if** $node$ is a leaf **then**
3: **return** $node$
4: **end if**
5: **switch** k **do**
6: **case** $k < k_0$
7: **return** LOOKUP(k, p_0)
8: **case** $k_i \leq k < k_{i+1}$
9: **return** LOOKUP(k, p_{i+1})
10: **case** $k_d \leq k$ ▷ Each node has at most $d \leq b$
11: **return** LOOKUP(k, p_{d+1})
12: **end procedure**
▷ **Step 1: Index Lookup**
13: $B' \leftarrow LOOKUP(L, B)$ ▷ Select a subset of postdiction models
▷ **Step 2: Recreate part of the Decayed Dataset using LSTM model**
14: **for all** $t \in R$ **do**
15: $pD' = RECREATE(B', t)$ ▷ Retrieve decayed data of specific time periods.
16: **end for**

for retrieving a selected subset from the decayed data, i.e., $pD' \subseteq D'$. For doing this, the *Recovery* algorithm inputs the set of models B as well as some spatiotemporal information L and R that will specify the amount of the decayed data to be retrieved. For example, L can be a cellular tower's location or a user's location associated to a cellular tower and R can be a range of timestamps, within which a number of records were generated and stored in D'. In any case, L and R will be utilized by the DP-tree *LOOKUP* function for deciding a subset of models $B' \subseteq B$ in line 13 that will be used for creating the pD' dataset in line 15.

Example: Consider the scenario of Figure 4 (Recovery Algorithm) where the data of cell tower *A* (part of cl_1) needs to be recovered for timestamps t_1,...,t_4. *LOOKUP* retrieves the LSTM model md_1 for cluster cl_1 created from all records related to cell towers *A* and *B* as shown in Step 1 of the given figure. In Step 2, the *Recovery* algorithm recreates the values of cell tower *A* for each timestamp t recovering in this way a part of the decayed data pD' using the selected LSTM model.

C. Performance Analysis

The secondary focus of *TBD-DP* is the efficient decaying of data and consequently the minimization of TBD storage space while maintaining a high accuracy during data recovery.

According to Definition 2.1 the total storage space S is equal to the actual data minus the decayed data based on f, plus any additional storage required by the decaying approach to achieve an optimal recreation of the decayed data. When there is no decaying $f = 0\%$ then $S = |D|+|B|$ (B could have been used for predicting future D values), which is the size of the actual (raw) data D and the size of the set of prediction models B. In the case of *TBD-DP*, $S = |D| - |D'| + |B|$, which is the actual data size minus the size of the decayed dataset $|D'| = |D| \times f\%$ plus the size of a set of models B, where $|D| >> |D'| + |B|$. When $f = 100\%$ then all data are decayed and the required storage space of *TBD-DP* is $S = |B|$. In the case of sampling, the storage space is equal to $S = |D| - |V|$, which is the actual data size minus a sample set $V = sampling(D', s)$ generated by sampling the decayed dataset D' with a pre-specified rate s. Note that $|D|-|D'|+|B| << |D|-|V|$ for a reasonable s that provides an *NRMSE* similar to *TBD-DP*.

According to Definition 2.2 the *NRMSE* measures the similarity of the decayed dataset D' and the recovered dataset pD'. Therefore, in cases where the decaying factor is $f = 0\%$, which corresponds to a low $|D'| = 0$ and no decaying is applied then *NRMSE* $= 0$ and when $f = 100\%$, which corresponds to a high $|D'| = |D|$ and all data are discarded then *NRMSE* $>> 0$. Moreover, it is reasonable to assume that in sampling, where a sample set V of the decayed data D' is permanently discarded with a sampling rate s then, its *NRMSE* (V, D') will be equal to the normalized difference between the sampled and the actual data. Finally, the *NRMSE* of the proposed *TBD-DP* will be equal to the normalized difference between the predicted data of the LSTM model and the actual data, i.e., *NRMSE* (pD', D').

IV. Prototype Description

We have developed a complete prototype architecture that integrates *TBD-DP* as part of the TBD Awareness project [1]. Our proposed architecture comprises of three layers (see Figure 2), namely Storage Layer, Indexing Layer and Application Layer.

The *Storage layer* passes newly arrived network snapshots through a lossless compression process storing the results on a replicated big data file system for availability and performance. This component is responsible for minimizing the required storage space with minimal overhead on the query response time. The intuition is to use compression techniques that yield high compression ratios but at the same time guarantee small decompression times. We particularly use GZIP compression that offers high compression/decompression speeds, with a high compression ratio and maximum compatibility with I/O stream libraries in a typical big data ecosystem we use.

[1]TBD Awareness, https://tbd.cs.ucy.ac.cy/

Fig. 5. The *TBD-DP* operator implemented inside the spatio-temporal SPATE architecture. The interface enables users to carry out high resolution visual analytics, without consuming enormous amounts of storage. The savings are quantified numerically with bar charts and visually with heatmaps.

Additionally, this layer uses the *TBD-DP* operator in order to provide the decay methods for the next layer. The storage layer is basically only responsible for the leaf pages of the *SPATE* index described in the next layer.

The Indexing Layer uses a multi-resolution spatio-temporal index, which is incremented on the rightmost path with every new data snapshot that arrives (i.e., every 30 minutes). In addition, the component computes interesting event summaries, called "highlights", from data stored in children nodes and stores them at the parent node. For each data exploration query, the internal node that covers the temporal window of the query is accessed, and its highlights are used to answer the query.

The Application Layer implements the querying module and the *data exploration* interfaces, which receive the data exploration queries in visual or declarative mode and use the index to combine the needed highlights and snapshots to answer the query. *SPATE* is equipped with an easy-to-use map-based web interface layer that hides the complexity of the system through a simple and elegant web interface.

V. Experimental Methodology and Evaluation

This section presents an experimental evaluation of our proposed *TBD-DP* operator. We start-out with the experimental methodology and setup, followed by two experiments. Particularly, in the first experiment, the performance of *TBD-DP* is compared against two baseline approaches and two decaying-based approaches with respect to various metrics on a set of anonymized datasets. The second experiment examines the influence of several control parameters on the performance of *TBD-DP*.

A. Methodology

This section provides details regarding the algorithms, metrics and datasets used for evaluating the performance of the proposed approach.

Testbed: Our evaluation is carried out on the DMSL VCenter IaaS datacenter, a private cloud, which encompasses 5 IBM System x3550 M3 and HP Proliant DL 360 G7 rackables featuring single socket (8 cores) or dual socket (16 cores)

Intel(R) Xeon(R) CPU E5620 @ 2.40GHz, respectively. These hosts have collectively 300GB of main memory, 16TB of RAID-5 storage on an IBM 3512 and are interconnected through a Gigabit network. The datacenter is managed through a VMWare vCenter Server 5.1 that connects to the respective VMWare ESXi 5.0.0 hosts. Computing Nodes: The computing cluster, deployed over our VCenter IaaS, comprises of 4 Ubuntu 16.04 server images, each featuring 8GB of RAM with 2 virtual CPUs (@ 2.40GHz). The images utilize fast local 10K RPM RAID-5 LSILogic SCSI disks, formatted with VMFS 5.54 (1MB block size). Each node uses Hadoop v2.5.2.

We utilize anonymized measurements from a real Telco operator that comprises of 1192 real cell towers (i.e., 3660 cells of 2G, 3G and LTE networks) distributed in an area of 5,896 km^2. The cells are connected through a Gigabit network to a datacenter. Each cell tower keeps several UMTS/GSM network logs for the performance of the tower and forwards the information through the base station controller (BSC) or the radio network controller (RNC) to be stored. There is a CDR server that generates call detail records (CDRs) for incoming and outgoing calls in the enterprise. When a CDR is generated in the CDR server, the management server and third-party application can use SFTP to obtain the CDR from the CDR server. Then the Telco can query the CDRs for call/data information and check outgoing call/data fees with the carrier.

Algorithms: The proposed *TBD-DP* operator is compared with the following approaches:

- **RAW:** does not apply any decaying on the whole dataset.
- **COMPRESSION:** the decayed dataset is compressed with the *GZIP* library, which has been shown in [5] to offer the best balance between compression/decompression speeds, compression ratios and compatibility with I/O stream libraries.
- **SAMPLING:** a sampling method that picks every second item in the input stream, yielding a 50% sample size.
- **RANDOM:** uniformly randomly select one record from the decayed dataset.

Note that RAW and RANDOM are the baseline approaches used to demonstrate the trade-off between the storage capacity and the NRMSE objectives.

Datasets: We utilize an anonymized dataset of telco traces comprising of $\sim$ 100M network measurements records (NMS) and 3660 cells (CELL) coming from 2G, 3G and LTE antennas. The data traffic is created from about 300K objects and has a total size of $\sim$10GB. We constructed 6 realistic datasets from real TBD obtained through *SPATE* described in Section V-A based on the *Key Performance Indicators (KPIs)* [16].

- **Calls (CS):** the number of calls ended normally during snapshot d_t.
- **Call Drops (CSD):** the number of calls dropped during snapshot d_t.
- **Handover Attempts (HA):** the amount of handovers into or from the cells attempted during a snapshot d_t.
- **Handovers (HS):** the number of successful handovers into or from the cells during a snapshot d_t.
- **Call Setup Attempts (CSA):** the amount of call setup processes attempted during snapshot d_t.
- **Call Setups (CE):** the amount of successful call setup processes during snapshot d_t.

Metrics: We evaluate the performance of *TBD-DP* using the metrics defined in Section II-A in all experiments:

- **Storage Capacity (S):** measures the total space that data and the DP-tree index occupy together, as a percentage of storage required by the RAW method (no decaying, no compression).
- **Normalized Root Mean Square Error (*NRMSE*):** measures the error of the recovered data D' using the NRMSE formula provided at the end of Section II. A lower *NRMSE* value indicates a higher accuracy in the recovered data.

Parameters: In all experiments the simulation parameters were configured as follows: (i) decay factor $f = 50\%$ (indicating the percentage on which we execute the LSTM); (ii) the ML model is LSTM and the number of neurons 16 x 16. The influence of each of those parameters on the proposed approach is investigated individually in Experiment 2 by fixing the rest of the parameters accordingly.

B. Experiment 1: Performance Evaluation

In the first experiment, we evaluate the performance of the proposed *TBD-DP* operator against all four algorithms and over all datasets introduced in Section V-A, with respect to space capacity (as a percentage to the RAW data) and accuracy (in terms of *NRMSE* on the decayed set of data).

Figure 6 clearly demonstrates the trade-off between the space capacity S and the *NRMSE* objectives on the results of the baseline approaches, since *RAW* (no decaying) approach obtained the worst possible $S = 100\%$ of the whole dataset, and the lowest error *NRMSE* $= 0$. In contrast, the *RANDOM* (almost all data are decayed) approach obtained the best possible $S = 50\%$ of the whole dataset and the worst *NRMSE* ≈ 100 on the decayed dataset, for a decaying factor $f = 50\%$. The results of the three other approaches appear in between the results of the two baseline approaches. The proposed *TBD-DP* operator, however, provides around 25% and 50% better space capacity S compared to *COMPRESSION* and *SAMPLING* approaches, respectively. This is due to the fact that the additional space required by the set of LSTM models is much less than the sample set of *SAMPLING* and the compressed decayed dataset of *COMPRESSION*.

In terms of *NRMSE*, the *TBD-DP* outperforms the *SAMPLING* approach by 50%, on average, in all datasets. The *COMPRESSION* approach provides an optimal *NRMSE* $= 0$, since it does not apply any prediction on the decayed data, but recovers them via decompression, when requested. The *COMPRESSION* approach however, can not be customized to achieve an even lower disk space occupancy. On the other hand, the *TBD-DP* can be configured, through its f parameter, to achieve a space occupancy that will fit the space budget of

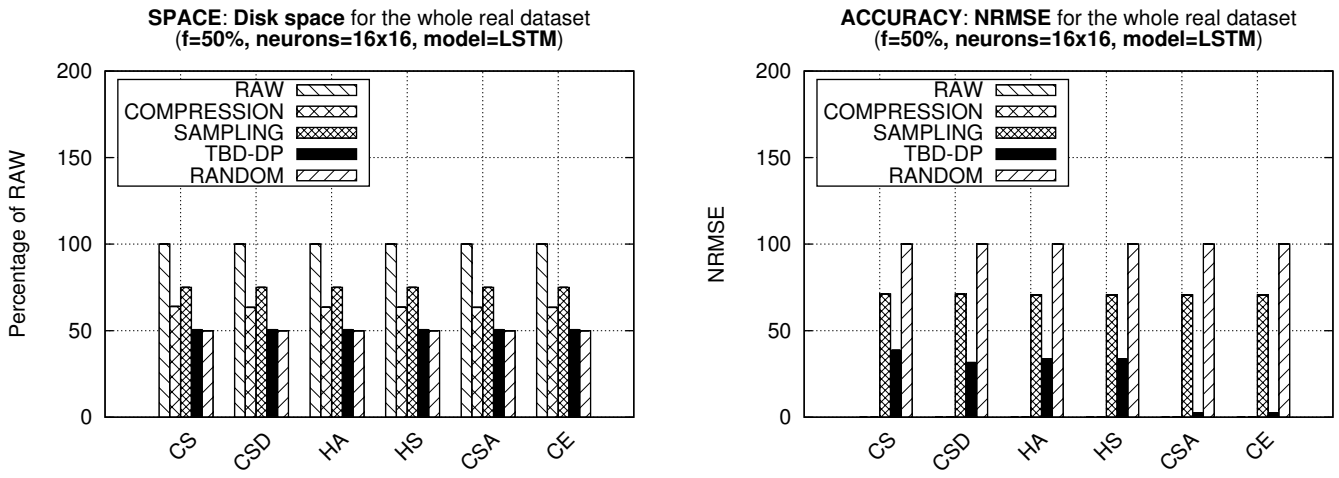

Fig. 6. Performance Evaluation: *TBD-DP* evaluation in terms of storage capacity S as a percentage to the RAW data (left) and accuracy in terms of *NRMSE* on the decayed set of data (right) in all datasets.

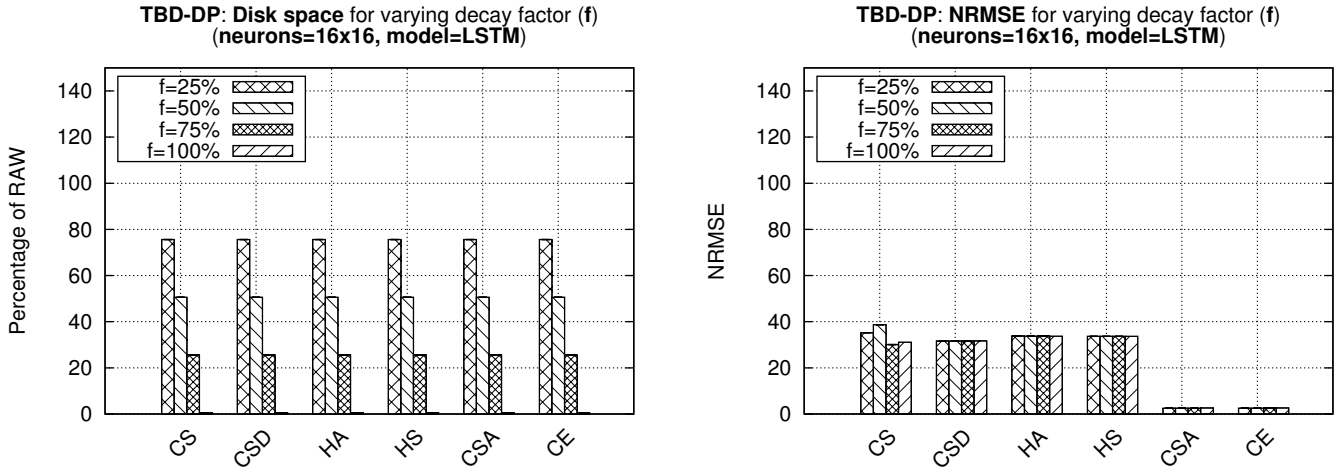

Fig. 7. Control Experiment - Decaying factor f: examining the storage capacity S and *NRMSE* of the proposed *TBD-DP* approach while varying f.

the application. This particular parameter will be investigated in the next experiment.

C. Experiment 2: Control Experiments

In Experiment 2, we examine the influence of several control parameters on the performance of the proposed *TBD-DP* in terms of S and *NRMSE*. Specifically, we vary the decay factor (f), the ML models and the number of neurons on LSTM.

Figure 7 shows how the decaying factor f, and consequently the amount of data that will be decayed and represented by LSTM models, affect the S and *NRMSE* of the proposed *TBD-DP* operator. The results show that the storage capacity required by the *TBD-DP* decreases as the decaying factor increases, which is reasonable due to the fact that the highest f is, the more data need to be decayed and therefore more disk space will be released. The accuracy of the proposed *TBD-DP* however, is not influenced, since *NRMSE* remains almost the same for all decaying factors, in most datasets. This shows the scalability and generalizability of the proposed approach, which is not influenced from the increase on the decaying dataset size. It is also important to note that the variations on the *NRMSE* obtained by *TBD-DP* between the datasets is mainly due to the different characteristics of each dataset.

Figure 8 examines the performance of the *TBD-DP* operator in terms of S and *NRMSE* when combined with three different ML models, namely, the traditional *Recurrent Neural Network (RNN)*, the *Gated Recurrent Unit (GRU)* [17] and the *Long Short Term Memory (LSTM)* that is finally adopted by the proposed approach. The results show that *TBD-DP* maintains a similar storage capacity for different learning models, with a slight increase (about 1%) when the LSTM model is used. In terms of *NRMSE*, however, the *TBD-DP*+LSTM combination clearly outperforms the other two combinations providing around 75% less error, on average.

Finally, Figure 9 examines how the number of neurons of the LSTM model influences the *TBD-DP*'s performance. The

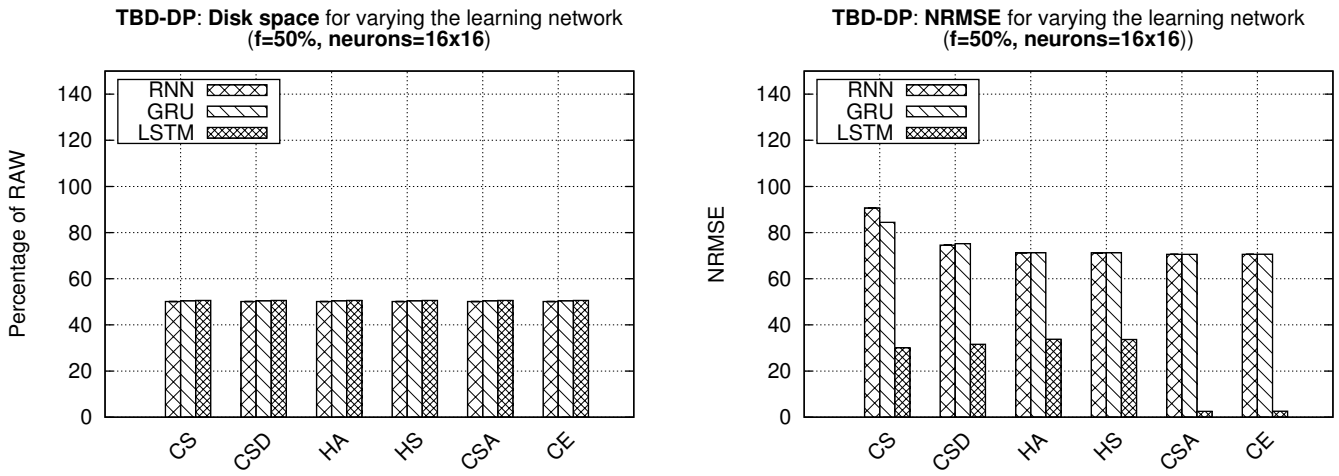

Fig. 8. Control Experiment - Learning Models: examining the storage capacity S and *NRMSE* of the proposed *TBD-DP* approach while combined with various ML models.

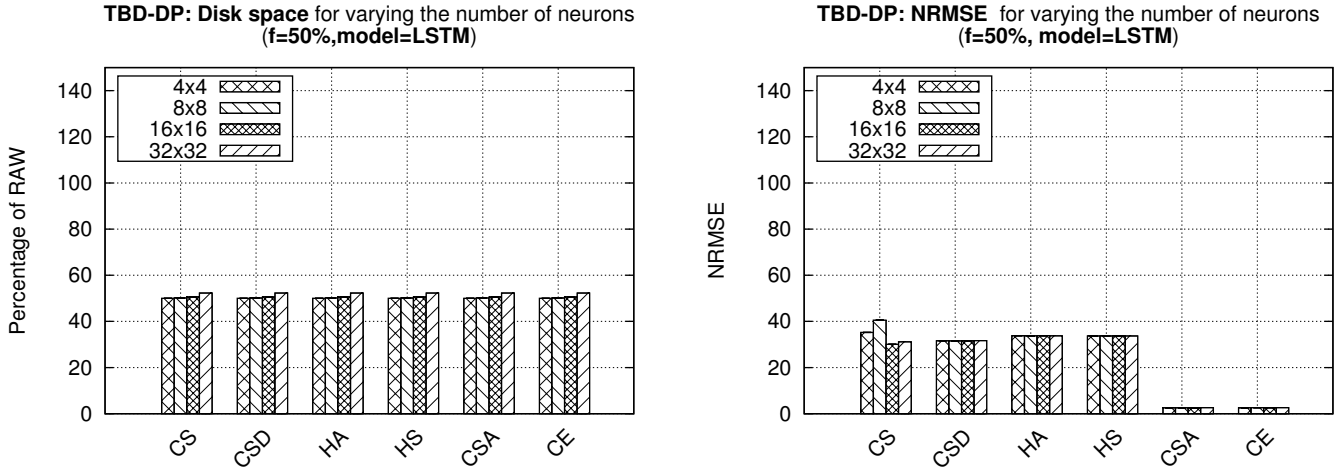

Fig. 9. Control Experiment - Number of neurons in LSTM: examining the storage capacity S and *NRMSE* of the proposed *TBD-DP* approach while varying the number of neurons in the LSTM.

results support our previous observations on the scalability and generalizability of the proposed *TBD-DP* approach. The increase on the number of neurons slightly influences the *TBD-DP* in terms of storage capacity, since the required space slightly increases. This is reasonable since the increase on the number of neurons results in "bigger" models that require more disk space to be stored. The additional required space, however, is almost negligible compared to the disk space needed to store the actual data before decaying. In terms of *NRMSE*, the increase on the number of neurons does not influence the performance of the *TBD-DP* operator, since *NRMSE* remains almost the same while varying this control parameter in almost all datasets.

VI. Related Work

In this section, we present related research work for completeness.

A. Telco Big Data (TBD) Research

Telco research can be roughly classified into the following three categories: (i) real-time analytics and detection; (ii) predicting user behavior; and (iii) privacy. There is also Telco research that focus on applications that Telcos can use to improve their services and revenue. Such kind of literature, however, is orthogonal to the topic of this article.

Real-time Analytics and Detection: Zhang et al. [1] have developed *OceanRT* for managing large spatiotemporal data, such as Telco OSS data, running on top of cloud infrastructure. It contains a novel storage scheme that optimizes queries with joins and multi-dimensional selections. Yuan et al. [18] present *OceanST* that features: (i) an efficient loading mechanism of ever-growing Telco MBB data; and (ii) new spatiotemporal index structures to process exact and approximate spatiotemporal aggregate queries in order to cope with the huge volume of MBB data. Iyer et al. [4] present *CellIQ* to optimize queries

such as "spatiotemporal traffic hotspots" and "handoff sequences with performance problems". It represents the snapshots of cellular network data as graphs and leverages on the spatial and temporal locality of cellular network data.

Braun et al. [19] developed a scalable distributed system that efficiently processes mixed workloads to answer event stream and analytic queries over Telco data. Bouillet et al. [20] proposed a system on top of IBM's InfoSphere Streams middleware that analyzes 6 billion CDRs per day in real-time. Abbasoğlu et al. [21] present a system for maintaining call profiles of customers in a streaming setting by applying distributed stream processing.

Experience, Behavior and Retention Analytics: Huang et al. [2] empirically demonstrate that customer churn prediction performance can be significantly improved with telco big data. Although BSS data have been utilized in churn prediction very well in the past decade, the authors show how with a primitive Random Forest classifier telco big data can improve churn prediction accuracy from 68% to 95%. Luo et al. [6] propose a framework to predict user behavior involving more than one million telco users. They represent users as documents containing a collection of changing spatiotemporal "words" that express user behavior. By extracting the users' space-time access records from MBB data, they learn user-specific compact topic features that they use for user activity level prediction.

Privacy: Hu et al. [22] study Differential Privacy for data mining applications over telco big data and show that for real-word industrial data mining systems the strong privacy guarantees given by differential privacy are traded with a 15% to 30% loss of accuracy. Privacy and confidentiality are critical for telcos' reliability due to the highly sensitive attributes of user data located in CDR, such as billing records, calling numbers, call duration, data sessions, and trajectory information.

B. Compressing Incremental Archives

Domain-specific compression techniques are often adopted for compressing spatiotemporal climate data [23], text document collections [24], scientific simulation floating point data [25]–[28], and floating point data streams [29]. Moreover, several research studies [30]–[32] have utilized differential compression techniques for studying the trade-off between compression ratio and decompression times for incremental archival data. None of these prior research works, however, has been proposed for dealing with data decaying in Telco-specific distributed systems.

C. Data Synopsis

Sampling refers to the process of randomly selecting a subset of data elements from a relatively large dataset. Sophisticated techniques, such as Bernoulli and Poisson sampling, choose data elements using probabilities and statistics. Chaudhuri et al. [33] proposed *stratified sampling* where the probability of the selection is biased. In order to encounter the big data sampling issue, Zeng et al. [34] implemented G-OLA, which is a model that generalizes online aggregation in order to support general OLAP queries utilizing delta maintenance algorithms. Particularly, BlinkDB [35] allows users to choose the error bounds and the response time of query using dynamic sampling algorithms. SciBORQ [36] is a framework that allows the user to choose the quality of the query result based on multiple interesting data samples called impressions.

Several works have adapted the sampling processes to create synopsis of data in order to achieve low response time for ad-hoc queries [36]. Data sketches [12] are compact data structures that enable to efficiently estimate the count of occurences in massive data (contrary to Bloom filters, it encodes a potentially massive number of item types in a small array). Additionally, Wei et al. proposed persistent sketches that can answer queries at any prior time [37] and have the ability to merge in order to answer a generalization query [38].

VII. Conclusions

In this paper, we present a novel decaying operator for Telco Big Data (TBD), coined *TBD-DP (Data Postdiction)*. *TBD-DP* relies on existing ML algorithms to abstract TBD into compact models that can be stored and queried when necessary. Our proposed *TBD-DP* operator has the following two conceptual phases: (i) in an offline phase, it utilizes a LSTM-based hierarchical ML algorithm to learn a tree of models (coined *TBD-DP* tree) over time and space; (ii) in an online phase, it uses the *TBD-DP* tree to recover data within a certain accuracy. In our experimental setup, we measure the efficiency of the proposed operator using a $\sim$10GB anonymized real telco network trace and our experimental results in Tensorflow over HDFS are extremely encouraging as they show that *TBD-DP* saves an order of magnitude storage space while maintaining a high accuracy on the recovered data.

In the future we aim to generalize data decaying operators beyond TBD into new domains (e.g., signals from other type of IoT). This task might give space to new ML algorithms. Additionally, we aim to theoretically derive the accuracy/efficiency bounds of our data postdiction framework. Finally, we plan to carry out an extensive experimental study that will focus solely on decaying of big data.

References

[1] S. Zhang, Y. Yang, W. Fan, L. Lan, and M. Yuan, "Oceanrt: Real-time analytics over large temporal data," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 1099–1102.

[2] Y. Huang, F. Zhu, M. Yuan, K. Deng, Y. Li, B. Ni, W. Dai, Q. Yang, and J. Zeng, "Telco churn prediction with big data," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD. New York, NY, USA: ACM, 2015, pp. 607–618.

[3] F. Zhu, C. Luo, M. Yuan, Y. Zhu, Z. Zhang, T. Gu, K. Deng, W. Rao, and J. Zeng, "City-scale localization with telco big data," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ser. CIKM. New York, NY, USA: ACM, 2016, pp. 439–448.

[4] A. P. Iyer, L. E. Li, and I. Stoica, "Celliq: Real-time cellular network analytics at scale," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'15. Berkeley, CA, USA: USENIX Association, 2015, pp. 309–322.

[5] C. Costa, G. Chatzimilioudis, D. Zeinalipour-Yazti, and M. F. Mokbel, "Efficient exploration of telco big data with compression and decaying," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, April 2017, pp. 1332–1343.

[6] C. Luo, J. Zeng, M. Yuan, W. Dai, and Q. Yang, "Telco user activity level prediction with massive mobile broadband data," *ACM Trans. Intell. Syst. Technol.*, vol. 7, no. 4, pp. 63:1–63:30, May 2016.

[7] C. Costa, G. Chatzimilioudis, D. Zeinalipour-Yazti, and M. F. Mokbel, "Towards real-time road traffic analytics using telco big data," in *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, BIRTE, Munich, Germany, August 28, 2017*, 2017, pp. 5:1–5:5. [Online]. Available: http://doi.acm.org/10.1145/3129292.3129296

[8] E. Savitz, "Forbes magazine," 2012, [Online; April 16, 2012]. [Online]. Available: https://goo.gl/eM1uwV

[9] C. LaChapelle, "The cost of data storage and management: where is the it headed in 2016?" 2016. [Online]. Available: http://www.datacenterjournal.com/cost-data-storage-management-headed-2016/

[10] M. L. Kersten and L. Sidirourgos, "A database system with amnesia." in *CIDR*, 2017.

[11] M. L. Kersten, "Big data space fungus," in *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*, 2015.

[12] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Found. Trends databases*, vol. 4, no. 1–3, pp. 1–294, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1561/1900000004

[13] D. Barbará, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. E. Ioannidis, H. V. Jagadish, T. Johnson, R. T. Ng, V. Poosala, K. A. Ross, and K. C. Sevcik, "The new jersey data reduction report," *IEEE Data Eng. Bull.*, vol. 20, no. 4, pp. 3–45, 1997. [Online]. Available: http://sites.computer.org/debull/97DEC-CD.pdf

[14] K. Krishna, D. Jain, S. V. Mehta, and S. Choudhary, "An lstm based system for prediction of human activities with durations," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 4, pp. 147:1–147:31, Jan. 2018. [Online]. Available: http://doi.acm.org/10.1145/3161201

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[16] J. Laiho, A. Wacker, and T. Novosad, *Radio Network Planning and Optimisation for UMTS*. John Wiley & Sons, 2006.

[17] R. Dey and F. M. Salemt, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2017, pp. 1597–1600.

[18] M. Yuan, K. Deng, J. Zeng, Y. Li, B. Ni, X. He, F. Wang, W. Dai, and Q. Yang, "Oceanst: A distributed analytic system for large-scale spatiotemporal mobile broadband data," *Proc. VLDB Endow.*, vol. 7, no. 13, pp. 1561–1564, Aug. 2014.

[19] L. Braun, T. Etter, G. Gasparis, M. Kaufmann, D. Kossmann, D. Widmer, A. Avitzur, A. Iliopoulos, E. Levy, and N. Liang, "Analytics in motion: High performance event-processing and real-time analytics in the same database," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: ACM, 2015, pp. 251–264.

[20] E. Bouillet, R. Kothari, V. Kumar, L. Mignet, S. Nathan, A. Ranganathan, D. S. Turaga, O. Udrea, and O. Verscheure, "Processing 6 billion cdrs/day: From research to production (experience report)," in *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, ser. DEBS '12. New York, NY, USA: ACM, 2012, pp. 264–267.

[21] M. A. Abbasoğlu, B. Gedik, and H. Ferhatosmanoğlu, "Aggregate profile clustering for telco analytics," *Proc. VLDB Endow.*, vol. 6, no. 12, pp. 1234–1237, Aug. 2013.

[22] X. Hu, M. Yuan, J. Yao, Y. Deng, L. Chen, Q. Yang, H. Guan, and J. Zeng, "Differential privacy in telco big data platform," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1692–1703, Aug. 2015.

[23] T. Bicer, J. Yin, D. Chiu, G. Agrawal, and K. Schuchardt, "Integrating online compression to accelerate large-scale data analytics applications," in *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*. IEEE, 2013, pp. 1205–1216.

[24] H. Yan, S. Ding, and T. Suel, "Inverted index compression and query processing with optimized document ordering," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 401–410.

[25] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Compressing the incompressible with isabela: In-situ reduction of spatio-temporal data," in *European Conference on Parallel Processing*. Springer, 2011, pp. 366–379.

[26] E. R. Schendel, Y. Jin, N. Shah, J. Chen, C.-S. Chang, S.-H. Ku, S. Ethier, S. Klasky, R. Latham, R. Ross *et al.*, "Isobar preconditioner for effective and high-throughput lossless data compression," in *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 138–149.

[27] E. Soroush and M. Balazinska, "Time travel in a scientific array database," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 98–109.

[28] S. Bhattacherjee, A. Deshpande, and A. Sussman, "Pstore: An efficient storage framework for managing scientific data," in *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, ser. SSDBM '14. New York, NY, USA: ACM, 2014, pp. 25:1–25:12. [Online]. Available: http://doi.acm.org/10.1145/2618243.2618268

[29] M. Burtscher and P. Ratanaworabhan, "Fpc: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, 2009.

[30] F. Douglis and A. Iyengar, "Application-specific delta-encoding via resemblance detection." in *USENIX Annual Technical Conference, General Track*, 2003, pp. 113–126.

[31] L. L. You, K. T. Pollack, D. D. Long, and K. Gopinath, "Presidio: a framework for efficient archival data storage," *ACM Transactions on Storage (TOS)*, vol. 7, no. 2, p. 6, 2011.

[32] S. Bhattacherjee, A. Chavan, S. Huang, A. Deshpande, and A. Parameswaran, "Principles of dataset versioning: Exploring the recreation/storage tradeoff," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1346–1357, 2015.

[33] S. Chaudhuri, G. Das, and V. Narasayya, "Optimized stratified sampling for approximate query processing," *ACM Trans. Database Syst.*, vol. 32, no. 2, Jun. 2007. [Online]. Available: http://doi.acm.org/10.1145/1242524.1242526

[34] K. Zeng, S. Agarwal, A. Dave, M. Armbrust, and I. Stoica, "G-ola: Generalized on-line aggregation for interactive analysis on big data," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: ACM, 2015, pp. 913–918. [Online]. Available: http://doi.acm.org/10.1145/2723372.2735381

[35] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "Blinkdb: Queries with bounded errors and bounded response times on very large data," in *Proceedings of the 8th ACM European Conference on Computer Systems*, ser. EuroSys '13. New York, NY, USA: ACM, 2013, pp. 29–42. [Online]. Available: http://doi.acm.org/10.1145/2465351.2465355

[36] L. Sidirourgos, Martin, and P. Boncz, "Sciborq: Scientific data management with bounds on runtime and quality," in *In Proc. of the Intl Conf. on Innovative Data Systems Research (CIDR*, 2011, pp. 296–301.

[37] Z. Wei, G. Luo, K. Yi, X. Du, and J.-R. Wen, "Persistent data sketching," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: ACM, 2015, pp. 795–810. [Online]. Available: http://doi.acm.org/10.1145/2723372.2749443

[38] P. K. Agarwal, G. Cormode, Z. Huang, J. Phillips, Z. Wei, and K. Yi, "Mergeable summaries," in *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, ser. PODS '12. New York, NY, USA: ACM, 2012, pp. 23–34. [Online]. Available: http://doi.acm.org/10.1145/2213556.2213562