Identifying and Estimating Persistent Items in Data Streams

Haipeng Dai Muhammad Shahzad Alex X. Liu Meng Li Yuankun Zhong Guihai Chen

***Abstract*—This paper addresses the fundamental problem of finding persistent items and estimating the number of times each persistent item occurred in a given data stream during a given period of time at any given observation point. We propose a novel scheme, PIE, that can not only accurately identify each persistent item with a probability greater than any desired false negative rate (FNR) but can also accurately estimate the number of occurrences of each persistent item. The key idea of PIE is that it uses Raptor codes to encode the ID of each item that appears at the observation point during a measurement period and stores only a few bits of the encoded ID in the memory. The item that is persistent occurs in enough measurement periods that enough encoded bits for the ID can be retrieved from the observation point to decode them correctly and get the ID of the persistent item. To estimate the number of occurrences of any given persistent item, PIE uses maximum likelihood estimation based statistical techniques on the information already recorded during the measurement periods. We implemented and evaluated PIE using three real network traffic traces and compared its performance with three prior schemes. Our results show that PIE not only achieves the desire FNR in every scenario, its average FNR can be 19.5 times smaller than the FNR of the adapted prior scheme. Our results also show that PIE achieves any desired success probability in estimating the number of occurrences of persistent items.**

***Index Terms*—Electromagnetic Radiation, Wireless Power Transfer, Optimization, Distributed Algorithm.**

I. Introduction

A. Motivation and Problem Statement

With the increase in the popularity of applications that process large volumes of data generated at high rates, such as internet traffic analysis [1], business decision support [2], direct marketing [3], and sensor data mining [4], data stream mining is becoming more and more important. Data stream mining is the process of extracting information from a data stream, where a data stream is an ordered sequence of data items that can often only be read once using limited computing and storage resources [5]. One of the heavily studied problems in data stream mining is the fundamental problem of mining frequent items, which deals with finding items that occur most frequently in a given data stream over a period of time [6]–[10]. Frequent item mining finds applications in a variety of practical scenarios such as finding most popular destinations or heaviest users on the Internet by treating packets as items [9], and finding most popular terms in queries made to an Internet search engine [11].

A more generalized version of frequent item mining is the *persistent frequent item* mining. A persistent frequent item, unlike a frequent item, does not necessarily occur more frequently compared to other items over a short period of time, rather *persists* and occurs more frequently compared to other items only over a long period of time. For the sake of brevity, onwards, we will call persistent frequent items just persistent items. Persistent item mining finds applications in a variety of settings such as network security and click-fraud detection. For network security, persistent item mining can be used to detect stealthy DDoS attacks, where an attacker does not overwhelm the target rather degrades its performance using a small number of attacking machines for a long period of time [12]. Similarly, it can be used to detect stealthy port scanning attacks, where an attacker uses a small probing rate for a long period of time to discover system vulnerabilities [12]. Persistent item mining can also be used to detect network bots by monitoring the communication between a bot and its C&C server [13]. For click-fraud detection, persistent item mining can be used to detect if automatic robots are periodically generating clicks on an ad to increase the payment for an advertiser in pay-per-click online advertising systems [14].

In this paper, we address the fundamental problem of finding, or in other words, *identifying* persistent items in a given data stream and *estimating* the number of times each persistent item occurs in that data stream during a given period of time at any given observation point. An *observation point* is any computing device that can see the data stream and can process and store information about items in the data stream. Examples of observation points include end hosts, network middleboxes (such as routers and switches), firewalls, and other computing devices. To formally state the problem, let us divide the given period of time into small equally sized measurement periods. Let us also define the term *occurrence* as an event that one or more instances of an item appear in a measurement period. Formally, *given a period of time comprised of T consecutive equally sized measurement periods, a threshold T_{th}, and a desired false negative rate, identify the persistent items,* i.e.*, the items that occur in more*

This work was supported in part by the National Natural Science Foundation of China under Grant 61502229, Grant 61373130, Grant 61672276, No. 61472184, and No. 61321491, in part by the Fundamental Research Funds for the Central Universities under Grant 021014380079, in part by the Key Research and Development Project of Jiangsu Province under Grant No. BE2015154 and BE2016120, and the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing University, in part by the National Science Foundation under Grant Numbers CNS-1616273 and CNS-CNS 1616317, and in part by the Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program. *(Corresponding author: Haipeng Dai.)*

H. Dai, A. X. Liu, Meng Li, Y. Zhong, and Guihai Chen are with the State Key Laboratory for Novel Software Technology, Nanjing University, China, Nanjing 210023 (e-mail: haipengdai@nju.edu.cn; alexliu@cse.msu.edu; menson@smail.nju.edu.cn; kun@smail.nju.edu.cn; gchen@nju.edu.cn).

M. Shahzad is with Department of Computer Science, North Carolina State University, USA (e-mail: mshahza@ncsu.edu).

than T_{th} out of the T measurement periods, such that the percentage of persistent item not successfully identified is less than the desired false negative rate Furthermore, given a confidence interval $\beta \in (0, 1]$, and a required success probability $\alpha \in [0, 1)$, estimate the value $\tilde{\mathcal{N}}_i$ of the number of occurrences $\mathcal{N}_i$ of each persistent item i, such that when $\mathcal{N}_i \geq T_{th}$, $P\{|\tilde{\mathcal{N}}_i - \mathcal{N}_i| \leq \beta\mathcal{N}_i\} \geq \alpha$.

B. Limitations of Prior Art

First, some related problem is the extensively studied frequent item mining problem [6]–[9]. Unfortunately, none of the existing schemes for mining frequent items can be directly used for mining persistent items because frequent item mining schemes count the frequencies of items without taking the temporal dimension into account. More specifically, a frequent item mining scheme can calculate the total number of instances in t measurement periods but it cannot calculate the number of measurement periods in which those instances occurred out of the t measurement periods. For example, in a stealthy DDoS attack, if an attacking machine sends $10,000$ packets during 100 measurement periods while a legitimate machine sends the same number of packets in 2 measurement periods, existing frequent item mining schemes can calculate that both machines sent $10,000$ packets each, but cannot determine the amount of time they took to send these packets. Furthermore, as existing schemes for mining frequent items do not take the temporal dimension into account, they cannot eliminate duplicate items within a measurement period, such as multiple packets belonging to the same flow, which can result in the over-counting of the number of occurrences of an item.

Second, the most related works regarding persistent item mining [15]–[17] are essentially based on sampling, which is fundamentally different from our scheme, and they need to record the whole ID information for all recorded items, which may not be space efficient compared with our scheme.

C. Proposed Approach

In this paper, we propose PIE, a Persitent items Identification and Estimation scheme. PIE cannot only accurately identify each persistent item i but can also accurately estimate the number of occurrences $\mathcal{N}_i$ of the persistent item i. For identification, PIE identifies each persistent item with a failure probability lower than the desired false negative rate (FNR), where FNR is defined as the ratio of the number of persistent items that PIE fails to identify to the number of all persistent items. The key idea of PIE is that it uses Raptor codes (proposed in [18]) to encode the ID of each item that appears at a given observation point during a measurement period and stores only a few bits of the encoded ID in the memory of that observation point during that measurement period. The ID of an item can be any arbitrary identifier such as for network IP packets, it can be the standard five tuple (*i.e.*, source IP, destination IP, source port, destination port, and protocol type). The motivation behind encoding the ID with Raptor codes and then storing the result instead of storing the original ID is that the Raptor codes encode the given ID into a potentially limitless sequence of bits. To decode the bits to obtain the original ID, Raptor codes need only a small subset of all the encoded bits. As we shall see later, it is possible that when an observation point tries to store information (either encoded or non-encoded) about ID of an item at a memory location, that memory location may already be occupied, resulting in a collision and loss of ID information. Had we been storing original non-encoded ID, the bits that the observation point could not store due to collision would be lost and the original ID may never be retrieved correctly. On the contrary, when storing bits for the ID of an item encoded with Raptor codes, even if some bits are lost due to collision during a few measurement periods, if the item is persistent, there will still be enough encoded bits stored during other measurement periods such that they can be correctly decoded to get the item ID. The motivation behind storing only a few bits of the encoded ID during each measurement period is to minimize the memory required at the observation point. The item that is persistent will occur in enough measurement periods that enough encoded bits for the ID can be retrieved from the observation point to decode them correctly and get the item ID. If we store enough encoded bits for each item during each measurement period such that the bits could be decoded to obtain the original ID from just a single measurement period, it would only result in wasted memory. To identify persistent items, PIE simultaneously processes the information of IDs stored during all measurement periods. For each persistent item, with a high probability, PIE obtains enough encoded bits to decode them correctly and get the ID of the persistent item. We have theoretically calculated the expression for FNR. PIE uses this expression to calculate the optimal number of encoded bits for each ID that it should store during the given measurement period such that the actual FNR never exceeds the desired FNR and at the same time, the amount of memory consumed at the observation point in storing the encoded bits is minimum. For estimation, PIE uses the information already recorded at the observation points during all measurement periods and applies maximum likelihood estimation based statistical techniques to estimate the number of measurement periods in which any given persistent item occurred.

D. Key Technical Challenges

In designing PIE, we faced several technical challenges, out of which, we list three here. The first technical challenge is to store the IDs of persistent items without requiring large amounts of memory at the observation point. To address this challenge, instead of storing original IDs of items during each measurement period, we use Raptor codes to encode the ID of each item and then store only a few bits of the encoded ID during each measurement period. To ensure that the original ID of each persistent item can be recovered with a high probability while consuming minimum amount of memory, we theoretically calculate the expression for FNR as a function of the length of Raptor codes, and then determine the minimum required number of encoded bits to be stored during each measurement period.

The second technical challenge is to accurately recover IDs of items using the encoded bits from multiple measurement periods. The encoded bits for the ID of an item in a measurement period are different from the encoded bits for the ID of that item in a different measurement period. Therefore,

PIE cannot determine which sets of encoded bits belong to the same ID. To address this challenge, PIE calculates a hash-print of each ID by applying a hash function to the ID and stores it along with the encoded bits during each measurement period in which the item with that ID appears. Unlike the encoded bits, the hash-print stays the same across measurement periods. PIE uses this hash-print to first group all sets of encoded bits across the T measurement periods that have the same hash-print and then recovers the ID using only the encoded bits in this group. It is possible that the hash-print of two or more different IDs may be the same. To address this challenge, after recovering an ID from a group, PIE applies a two-step verification and accepts the recovered ID only of the ID passes the two tests.

The third technical challenge is to estimate the occurrences of any given persistent item from the recorded information. To address this challenge, we first calculate the number of measurement periods from the recorded information in which we observed the given item without a collision. We show that the number of such measurement periods can be modeled with a binomial distribution. Next, we derive the probability distribution of the total number of occurrences of the persistent item, formulate the occurrence estimation problem as a maximum likelihood estimation problem (MLE), and utilize traditional MLE approach for estimation.

E. Key Contributions

PIE brings forward the state of the art in persistent item identification and estimation on the following three fronts: reliability, scalability, and robustness. For reliability, PIE takes the desired FNR, success probability α, and confidence interval β as inputs from the system operator and ensures that the actual FNR in identifying the persistent items is smaller than the desired FNR and the estimate of the number of occurrences of any given persistent item satisfies the success probability and confidence interval requirements. For scalability, PIE stores information about all items in a single data structure during one measurement period regardless of the number of different items and the number of times each item appears. For robustness, PIE is robust to accidental events, such as loss of information due to SRAM failures, because the use of Raptor code based encoding makes PIE inherently robust to losing some encoded bits, and still be able to decode the available bits to correctly recover IDs.

We extensively evaluated PIE and compared it with IBF [19], [20] and CM sketch [7], which we adapted to enable persistent item identification, and Small-Space. In our evaluations, we used three real-world network traffic traces including a backbone traffic trace [21], an enterprise network traffic trace [22], and a data center traffic trace [23]. Our results also show that the actual FNR of PIE is always less than the desired FNR. Our results further show that PIE achieves, on average, 19.5 times lower FNR compared to the adapted IBF scheme. Similarly, our results show that PIE achieves at least 426.1 and 1898.2 times lower FPR compared to the adapted CM sketch and Small-Space, respectively. Our results also show that PIE always achieves any required success probability in estimating the number of occurrences of any given item.

II. Related Work

A. Frequent Items Identification

A large body of works has been devoted to identifying frequent items, and some of the existing schemes can be adapted to identify persistent items. The frequent items problem refers to finding the items that occur most frequently in a given data stream of items. There are two main classes of algorithms for finding frequent items: counter-based algorithms and sketch-based algorithms [9].

Counter-based Algorithms: Manku *et al.* proposed the Lossy Counting (LC) algorithm to find frequent items [6]. The LC algorithm maintains a tuple of a lower bound on the count and the difference between the upper bound and the lower bound for each item. For each item, LC either increments its count by 1 if the item was previously seen, or adds it into the memory if the item is new and appropriately sets the lower bound for it. LC algorithm does not have any false negatives and its false positives are bounded, where the bound is proportional to the size of data stream. Metwally *et al.* proposed the Space Saving (SS) algorithm, which stores k (item, count) pairs [8]. If the incoming item matches an item in one of the k pairs, SS increments the count in the corresponding pair by one. Otherwise, it replaces the pair with the smallest count value by a new pair, which contains the incoming item and the count equal to the smallest count value incremented by one. Like LC algorithm, SS algorithm also does not have any false negatives and its false positives are bounded, where the bound is proportional to the size of data stream. These two algorithms and other such counter-based algorithms need to keep a set of items and counters. Thus, their errors grow proportionally with the size of the data stream, which is unacceptable for streams with a large number of frequent items or with huge size.

Sketch-based Algorithms: Sketch-based algorithms use sketches, which essentially contain linear projections of the input. Charikar *et al.* proposed Count (C) sketch [24] that consists of a two-dimensional array of counters. For each input item, C-sketch maps it to a counter in each row of counters using a hash function. In each mapped counter, C-sketch adds a value $\in \{-1, +1\}$ using another hash function. The total space, time per update, and error are determined by the width and height of the array. Cormode *et al.* proposed the famous Count-Min (CM) sketch that also consists of a two-dimensional array of counters [7]. For each input item, CM-sketch maps it to a counter in each row of counters using a hash function and increments each of the mapped counter by 1. To estimate the number of times any given item appeared, CM-sketch first uses the hash-function to identify the counters in each row that the item maps to, then returns the value of the smallest counter as the estimate. CM-sketch consumes less space than C-sketch for a given error requirement. Both C- and CM-sketch can be extended to identify persistent items by allocating enough memory to store information about the ID of all items. Unfortunately, the amount of memory that these two algorithms require becomes prohibitively large when the number of items in a data stream is large. A common shortcoming of both counter-based and sketch-based algorithms is that as they do not take the temporal dimension into account,

they cannot eliminate duplicate items within a measurement period, such as multiple packets belonging to the same flow, which can result in the over-counting of the number of occurrences of an item for our persistent item problem. For this, one can maintain a conventional Bloom filter as described later in Section VII-A.

B. Persistent Items Identification

There exists some literature studying on persistent items identification [15]–[17]. Lahiri *et al.* proposed a small-space algorithm to approximately track persistent items over a large data stream [15], [16]. Specifically, for tracking persistent items over a fixed window, the algorithm establishes a hash-based filter in each measurement period, and lets each stream item sent through this filter. If an item is selected by the filter, then the whole item ID is recorded and the persistence of the item is tracked in future periods. Further, they investigated the setting of sliding windows under which only the substream of elements that belong to the n most recent timeslots are considered. In addition, Singh *et al.* studied the persistent items monitoring problem in distributed data streams, which is quite different from our problem in centralized data streams [17]. They applied a hash-based random sampler to reduce the number of tracked items, adopted a distributed distinct counting algorithm to reduce the memory and communication cost. Essentially, the above two algorithm are based on sampling, which is fundamentally different from our scheme, and they need to record the whole ID information for all recorded items that may not be space efficient compared with our scheme. Besides, we studied the persistent items identification problem in the conference version of this paper [25]. This paper extends the work in [25] by considering the persistent items estimation problem.

Next, we describe an orthogonal class of work, namely Invertible Bloom Filter (IBF), that is not designed for finding frequent items, but can be modified to identify persistent items. An invertible Bloom filter (IBF) is similar to a conventional Bloom filter except that it can be inverted to yield the IDs of some of the inserted items [19], [20]. An IBF contains an array of cells, which in turn contains three fields: idSum, hashSum, and count. For any incoming item, the item is mapped into several cells using hash functions. For each cell that the item is mapped to, the count is incremented by 1, the stored idSum is XORed with the item ID, and the stored hashSum is XORed with the hash of the item ID. In the decoding phase, the "pure" cells, *i.e.*, the cells with count field either 1 or -1, are first identified and the item IDs from them are recovered. Using the IDs recovered from these pure cells, IBF further decodes the IDs from other cells to which the recovered IDs were mapped to. Note that, IBF cannot always decode all IDs, *i.e.*, it has false negatives. IBF can be adapted for identifying persistent items by assigning one IBF to each measurement period and maintaining an additional Bloom filter as described earlier in Section VII-A. Unfortunately, this adapted IBF scheme has a very low memory efficiency because it needs to store the whole ID information for every item during every measurement period, unlike PIE, which stores only a few encoded bits of every ID during any given measurement period in which the item appeared.

III. Persistent Item Identification

To identify persistent items, PIE operates in two phases: recording phase and identification phase. In the recording phase, PIE records information about the ID of each item seen at the observation point during the given measurement period. To store this information, during each measurement period, PIE maintains a data structure called Space-Time Bloom Filter (STBF) in SRAM and records information about the items in this data structure. At the end of each measurement period, PIE transfers this data structure to the permanent storage for later analysis and starts a new instance of STBF in the next measurement periods. At the end of T measurement periods, PIE has T instances of STBF stored in its memory. In the identification phase, PIE analyzes these T instances of STBF to identify persistent items during the T measurement periods. Next, we first describe STBF and then explain PIE's recording and identification phases. We summarizes the notations and symbols used throughout this paper in Appendix.

A. Space-Time Bloom Filter

STBF can be regarded as an extended version of the conventional Bloom filter (BF) [26], which not only records the membership information of items in a set but also records information about their IDs. Let $h_1(.), \cdots, h_k(.)$ be k independent hash functions with uniformly distributed outputs. Given a set of elements S, BF first constructs an array A of m bits, where each bit is initialized to 0, and for each item e in S, BF sets the k bits $A[h_1(e)\%m], \cdots, A[h_k(e)\%m]$ to 1. To process a membership query of whether item e is in set S, BF returns true if all corresponding k bits are 1 (*i.e.*, returns $\wedge_{y=1}^{k} A[h_y(e)\%m]$). In STBF, each bit in BF is replaced by a *cell*. Thus, STBF is an array C_i of m cells. For each item e that appears during a measurement period i, PIE applies the same k independent hash functions $h_1(.), \cdots, h_k(.)$ on e to identify the cells $C_i[h_1(e)\%m], \cdots, C_i[h_k(e)\%m]$ in the array C_i of the STBF in this measurement period to which this item e maps. Each cell is comprised of three data fields: flag field, Raptor code field, and hash-print field. The Raptor code field of cell x, where $1 \leq x \leq m$, stores an r-bit Raptor code of the ID of an item e if some hash function maps the item e to this cell, *i.e.*, if for some y, $h_y(e)\%m = x$, where $1 \leq y \leq k$. Typically r is much smaller than the length of item IDs. PIE selects the value of r such that the probability of decoding the IDs of persistent flows is high during the identification phase, while at the same time the amount of memory Raptor code fields consume in the STBF is minimum. We will present the mathematical model that PIE uses to calculate the optimal value of r in Section V-B. PIE uses the information stored in the Raptor code fields of cells to recover the exact IDs of persistent items during the identification phase. We represent the Raptor code field of cell x with $C_{iR}[x]$. The hash-print field of cell x stores an p-bit hash-print of the ID of an item e if some hash function maps the item e to this cell, *i.e.*, if for some y, $h_y(e)\%m = x$, where $1 \leq y \leq k$. The hash-print of an item ID is calculated using a hash function, which is different from the k hash functions used to map the item to k cells in STBFs. PIE uses the hash-print values stored in the hash-print fields of cells during the identification phase

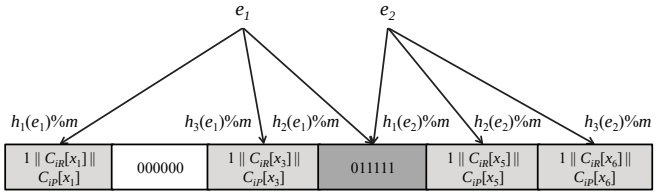


Fig. 1. Space-time code Bloom filter

to group the cells across different STBFs that are storing the Raptor codes for the same ID. We represent the hash-print field of cell x with $C_{iP}[x]$. PIE selects the value of p such that the probability of collision between hash-prints of different IDs is small, while at the same time the amount of memory hash-print fields consume in the STBF is minimum. We will present the mathematical model that PIE uses to calculate the optimal value of p in Section V-B. The flag field is just a single bit that indicates whether the cell is empty, singleton, or collided depending on whether the hash functions of none, one, or more than one items, respectively, mapped them to this cell. We represent the flag field of cell x with $C_{iF}[x]$. More specifically, $C_{iF}[x] = 1$ indicates that only one item is mapped to cell x, *i.e.*, cell x is a singleton cell. If $C_{iF}[x] = 0$ and $C_{iR}[x] = C_{iP}[x] = 0$, this indicates that no item is mapped to cell i, *i.e.*, cell x is an empty cell. If $C_{iF}[x] = 0$ and $C_{iR}[x] = 2^r - 1$ and $C_{iP}[x] = 2^p - 1$, *i.e.*, all bits of Raptor code field and hash-print field are set to 1, this indicates that more than one item are mapped to cell x, *i.e.*, cell x is a collided cell.

Figure 1 shows a simple toy example of STBF. The illustrated STBF has 6 cells, and applies 3 different hash functions to each item. For the two distinct items e_1 and e_2, STBF maps them to three cells each. There is one cell, the fourth from left, which both items are mapped to and is thus collided. There is one cell, the second from the left, which no item mapped to and is thus an empty cell with all bits set to 0. All remaining cells have the flag fields set to 1, and the appropriate Raptor codes and hash values stored.

Note that we chose to extend BF to build STBF. Another data structure that provides similar functionality but better performance compared to BF is Cuckoo filter [27]. Next, we explain why we did not choose to extend Cuckoo filter to build STBF. Cuckoo filter maintains an array of buckets and maps each item to two buckets using two independent hash functions. For any given item, if either of the two buckets the item maps to is empty, Cuckoo filter inserts the item in that bucket. If neither of the two buckets is empty, then Cuckoo filter kicks out the existing item from one of the two buckets, inserts the new item into this vacated bucket, and tries to re-insert the kicked-out item to the other bucket it maps to. Cuckoo filter repeats this process until either a maximum number of kicks are made or the kicked out item finds an empty bucket. Unfortunately, these kicks and reinsertions can take a long time and still result in insertion failures. If we were to use Cuckoo filters, whenever insertion of a given packet takes a larger duration of time due to these kicks, some of the packets arriving immediately after the given packet cannot be processed in time and their information will not be recorded by Cuckoo filter. This indeed happens in reality because the inter-arrival time of packets is comparable with the processing time of recording [28]. As the insertion time of Cuckoo filter is randomly distributed, any packet has similar probability to be missed. Thus, persistent items, which already have very few packets in each measurement period, will also be missed. In contrast, traditional BF has deterministic insertion time and can, therefore, be implemented to process packets at line-rate. As a result, we chose to extend BF to build STBF instead of extending Cuckoo filter. Besides, there also exist a number of sketch data structures (*e.g.*, [29]–[35]), but they are designed for different application scenarios and are not suitable for our considered persistent item identification scenario.

Algorithm 1: Recording Algorithm of PIE

Input: Items in the measurement period i
Output: A STBF recording item ID information

Initialize a STBF by setting all three fields (flag, Raptor code, and hash) in all cells of the STBF to 0;
while *an item e arrives* **do**
  Compute an r-bit Raptor code of the ID of the item and a p-bit hash-print of the ID;
  Calculate the k hash functions $h_y(e)$, where $1 \leq y \leq k$ and identifies the k cells $C_i[h_y(e)\%m]$ to which the element maps;
  for *each cell $C_i[h_y(e)\%m]$* **do**
    if *the cell is empty* **then**
      Set the flag field $C_{iF}[h_y(e)\%m]$ to 1, insert the Raptor code of the ID in $C_{iR}[h_y(e)\%m]$, and the hash-print of the ID in $C_{iP}[h_y(e)\%m]$;
    else if *the cell is singleton* **then**
      if *the Raptor codes and hash-print previously stored in the cell do not match those of the current item e* **then**
        Set the flag field to 0 and all the remaining bits in the cell to 1, *i.e.*, $C_{iF}[h_y(e)\%m] = 0$, $C_{iR}[h_y(e)\%m] = 2^r - 1$, and $C_{iP}[h_y(e)\%m] = 2^p - 1$;
Dump the recorded information in the STBF into the permanent storage.

B. Recording Phase

In the recording phase, PIE records information about the ID of each item seen at the observation point during the given measurement period. We show the details of the recording algorithm in Algorithm 1. Generally, PIE computes the Raptor code and hash-print of a coming item e and maps it to k cells. Then, PIE checks whether the mapped cells are empty, singleton, or collided, and accordingly modifies the stored information there. Next, we detail the process of Raptor code based encoding and hash-print calculation.

1) Raptor Code Based Encoding: Raptor code is a forward error correction (FEC) technology that is primarily used to provide application-layer protection against network packet loss [18]. The main advantages of Raptor codes are linear time encoding and decoding, the ability to encode a number of symbols into a potentially limitless sequence of symbols, and quite small decoding failure probability, P_{df}, which is the probability that one cannot decode the symbols encoded through Raptor codes to obtain the original set of symbols.

Let the length of the item ID be l bits, and let we use r bits to encode the ID using Raptor codes, P_{df} is given by the following equation (derived in [18]).

$$P_{df}(r;l) = \begin{cases} 1, & \text{if} \quad r < l \\ 0.85 \times 0.567^{r-l}, & \text{if} \quad r \geq l \end{cases} \tag{1}$$

In this paper, we use traditional Raptor code, which operates over Galois field $GF(2)$. There is an enhanced version of Raptor code called RaptorQ code [36], which operates over $GF(256)$ and achieves better performance. Unfortunately, RaptorQ code is not suitable for our setting for two reasons. First, it is space inefficient. Second, its output has a byte level granularity, while the output of $GF(2)$ Raptor code has bit-level granularity. Typically, the length of item IDs is no more than a few bytes. As the measurement periods are expected to be at least a few dozens, the required length of Raptor codes in each measurement period is just a few bits, which makes, $GF(2)$ Raptor code more suitable than RaptorQ code. Onwards, whenever we say Raptor code, we mean $GF(2)$ Raptor code.

The generalized encoding process based on Raptor codes is complicated, but in our setting, it can be viewed as a weighted linear sum of the bits in the item ID. Let us represent the ID of an element e with a vector $\mathbf{I}^e = (I_1^e \ I_2^e \ \ldots \ I_l^e)$, where I_x^e represents the x^{th} bit in the ID $\mathbf{I}^e$. To encode the IDs of items during any measurement period i, at the start of the period, PIE uses i as a seed to generate r encoding-coefficient vectors $\mathbf{a}_{ij}$, where $1 \leq j \leq r$. Clearly, $\mathbf{a}_{ij}$ can be generated and stored in advance to reduce the time overhead during recording, and it can be also precisely regenerated by using the seed i for decoding. Each encoding-coefficient vector $\mathbf{a}_{ij}$ contains l encoding-coefficients a_{ij}^x, where $1 \leq x \leq l$ and $a_{ij}^x \in \{0, 1\}$. The encoding-coefficients a_{ij}^x serves as the weight for the x^{th} bit in the weighted linear sum of the bits of the item ID. Note that PIE uses same r encoding-coefficients vectors for each item in a given measurement period, but the encoding-coefficient vectors for different measurement periods are different. Moreover, the operation of summing is performed over the Galois field of two elements, *i.e.*, $GF(2)$, and its result must be a binary value.

In a measurement period i, to generate the j^{th} of the r encoded bits of item ID e represented by R_{ij}^e, which will be stored in the Raptor code field of appropriate cells, *i.e.*, in $C_{iR}[h_y(e)\%m]$, where $1 \leq y \leq k$, PIE takes the dot product of the encoding-coefficient vector $\mathbf{a}_{ij}$ with the ID vector $\mathbf{I}^e$. PIE obtains this dot product for all values of j, where $1 \leq j \leq r$, to get the r bits of the Raptor code of the given item ID. This process is represented by the following equation.

$$\begin{pmatrix} a_{i1}^1 & a_{i1}^2 & \ldots & a_{i1}^l \\ a_{i2}^1 & a_{i2}^2 & \ldots & a_{i2}^l \\ \vdots & \vdots & & \vdots \\ a_{ir}^1 & a_{ir}^2 & \ldots & a_{ir}^l \end{pmatrix} \cdot \begin{pmatrix} I_1^e \\ I_2^e \\ \vdots \\ I_l^e \end{pmatrix} = \begin{pmatrix} R_{i1}^e \\ R_{i2}^e \\ \vdots \\ R_{ir}^e \end{pmatrix} \tag{2}$$

Note that PIE needs the IDs of all items to be of equal length, which is trivial because IDs with smaller length can be made equal in length to other IDs by appending appropriate number of 0s at the start of the ID.

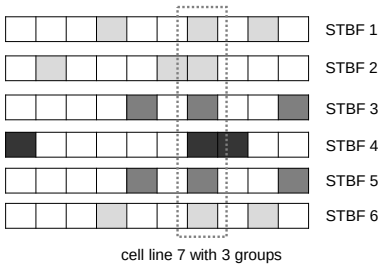


Fig. 2. Visual representation of a cell line

2) Hash-print Calculation: The process of calculating the hash-print is straight forward. PIE simply applies a hash function with uniformly distributed output to calculate the p-bit hash-print of any given ID and stores it in the hash-print field of the k appropriate cells. Note that, while Raptor codes of a given ID are different across measurement periods, *i.e.*, R_{ij}^e is different for different values of i, the hash-print value stays the same.

Lemma III.1. *The computational overhead of the recording algorithm is* $O(r \cdot l + k \cdot (r + p))$.

Proof. Please see Appendix for the proof. □

Apparently, choosing a small value of r, k and p will reduce the computational overhead, but this will downgrade the identification accuracy.

C. Identification Phase

In the identification phase, to recover the IDs of the persistent items with more than T_{th} occurrences in any set of T measurement periods, PIE processes the STBFs generated during those T periods and recovers the IDs of all such persistent items with high probability. In Section IV-A, we will calculate the probability that PIE fails to recover the ID of any given persistent item.

We show the details of the identification algorithm in Algorithm 2 (see Appendix). Generally, it contains three parts as stated below.

1) Processing Cell Lines: First, we process the so-called cell lines from Step 2 to Step 6. The motivation behind this type of grouping is that the cells containing the same hash-print are highly likely to contain the encoded bits of the ID of the same item because probability that hash-prints of two IDs will be the same is very small (but not zero; and we will shortly handle the case if hash-prints of multiple IDs indeed are the same). PIE uses the bits in the Raptor code fields of the cells in each group to recover the ID from that group. Figure 2 shows 6 STBFs obtained in 6 measurement periods, and the highlighted group of cells at the 7^{th} position constitute the cell line with $x = 7$. Also in Figure 2, each item is mapped to 3 locations, *i.e.*, $k = 3$. To keep the example simple, we have assumed that in this particular example, each STBF contains information about only one item. The same shade of grey in this figure represents the information with same hash-prints (but may be from different items). In this example, cell line with $x = 7$ has 3 different groups marked by 3 different shades from 4 items. Note that the ID of the item in STBF 2 differs from that in STBF 1 and 6 as their three cell locations are not exactly the same. This leads to mingled groups as will be

described in Section III-C3. Note that after processing all the cell lines, each group of cells has a unique hash-print.

2) Recovering Item IDs: We try to recover item IDs from Step 7 to Step 15. Generally, PIE uses Equation (2) to obtain $g \times r$ linear equations. Note that the value of r is much smaller than the ID length l and to solve the linear equations successfully to recover $\mathbf{I}^e$, PIE needs at least l equations, *i.e.*, PIE needs $g \times r$ to be $\geq l$, or in other words, it needs $g \geq \lceil l/r \rceil$ to be able to recover the original ID of the item of that group. PIE calculates the values of r and m such that the number of cells g in the group of a persistent item, *i.e.*, the item with more than T_{th} occurrences in T periods, is greater than $\lceil l/r \rceil$ with a very high probability. We will present the mathematical equations that PIE uses to calculate the values of r and m in Section V-B. Then, it verifies that the recovered ID is correct from Step 12 to Step 15. If the recovered ID has passed the second test, and can be considered correct with high probability.

3) Processing Mingled Groups: Moreover, we describe how PIE recovers the IDs if the cells in a group contain encoded bits coming from two or more IDs instead of a single ID. We call such groups *mingled groups*. This happens when the hash-prints of two or more IDs are the same, *i.e.*, there is a hash-print collision. For instance, cell line 7 in Figure 2 indeed has 4 groups, but two groups among them are mingled groups. In such a setting, when PIE recovers an ID, the hash-print of the recovered ID will not match the hash-print stored in the hash-print fields of the cells in the group. In this case, PIE temporarily removes g_r ($1 \leq g_r \leq g_T$, where g_T is a preset threshold) cells from the group and tries to recover the ID using only the remaining $g - g_r$ cells as shown from Step 17 to Step 21. Note that although PIE can set the threshold g_T to be equal to g, it typically does not do that because the value of g can be quite large, resulting in a prohibitively large number of iterations. PIE typically sets the threshold g_T to be no more than 2. Our evaluation results show that PIE recovers, on average, 93.5% of persistent items using this threshold.

Lemma III.2. *The computational overhead of the identification algorithm is* $O(mTl^3\binom{kT}{g_T})$.

Proof. Please see Appendix for the proof. □

Obviously, choosing a small value of g_T will substantially reduce the computational cost, but this comes at the cost of higher false negative rate.

Moreover, we will compare our algorithm to Invertible Bloom Filter (IBF) [19], [20], Count-Min (CM) sketch [7], and Small-Space [15], [16] in performance evaluation. Theoretically, the computational overhead of identification, which can be also explained as the decoding time, for both CM sketch Small-Space is a constant value, and that of IBF is $O(m)$ [19], [20]. Although our algorithm has the worst performance, the identification phase generally is done offline and has no rigid performance requirement as that for recording phase in typical applications, and thus we believe it is not the primal concern for implementation of our algorithm.

For the memory cost, if FNR and FPR are required to remain at a required low level, we expect that at each measurement period the space taken for each distinct item by our algorithm is $O(\frac{l}{T}+1)$ because the encoded bits with length proportional to the item ID length l is dispersed into T measurement periods and meanwhile the number of recorded bits for each item is at least $O(1)$ at each period. Therefore, in the worst case where all N distinct items appear at each measurement period, the total memory cost is $O((\frac{l}{T}+1)N)$. We note that the total memory cost for the permanent storage is $O((\frac{l}{T}+1)N)\times T = O(l+T)N)$ as it accepts all the dumped STBFs for all periods, but it is not a big issue as offline permanent storage is typically assumed to be sufficiently large; in contrast, the memory cost for online recording at each measurement period is our first concern as in practice the storage component used for such high-speed measurement usually has very limited memory size.

IV. Analysis of Persistent Item Identification

In this section, we derive the false negative rate (FNR) and false positive rate (FPR) of PIE in identifying persistent items. FNR is the ratio of the number of persistent items whose IDs PIE fails to recover to the number of all persistent items. FPR is the ratio of the number of non-persistent items that PIE declares as persistent to the number of all non-persistent items.

A. False Negative Rate

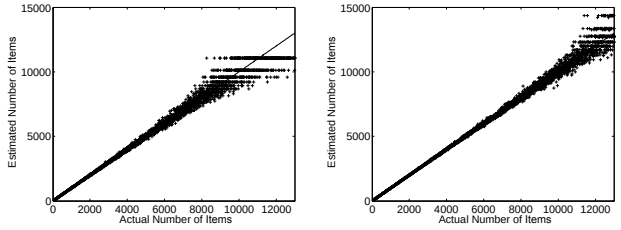
There are two reasons that can cause PIE to fail to recover the IDs of persistent items: hash-mapping collisions and hash-print collisions. Next we describe these two reasons.

Hash-mapping Collisions: It is possible that several cells out of the k cells $C_i[h_1(e)\%m], \cdots, C_i[h_k(e)\%m]$ to which a given item e is mapped using the k hash functions $h_1(e), \cdots, h_k(e)$ are collided because the k hash functions may map other items to these cells, especially if the number of items is large. Due to such hash-mapping collisions, the number of cells g in the group of item e during the identification phase may not satisfy the requirement $g \geq \lceil l/r \rceil$, leading to the failure of PIE in recovering the ID of the item e.

Hash-print Collisions: It is possible that due to the limited number of bits in the hash-print fields, the hash-prints of some items may be the same. Due to such hash-print collisions, PIE may put the cells of different items into the same group during the identification phase, *i.e.*, the group becomes *mingled*. Although PIE can recover IDs of multiple items from mingled groups as explained in Section III-C3, if the number of mingled cells is greater than the threshold g_T, PIE may fail to recover the IDs of any item from the group.

Next, we first derive the probabilities that PIE can recover the ID of any given item despite hash-mapping collisions and hash-print collisions. We name these probabilities "hash-mapping collision survival probability" P_m and "hash-print collision survival probability" P_p. After that, we use P_m and P_p to derive an expression for the overall FNR of PIE.

1) Hash-mapping Collision Survival Probability: Let the total number of occurrences of all items during T measurement periods be N. Thus, the expected number of occurrences in each measurement period is N/T. Consequently, the probability P_{nc} that a cell in a given STBF to which a given item e maps is not collided, *i.e.*, no other item maps to this cell is given by the following equation.



(a) Linear counting method (b) Our method

Fig. 3. Comparison of our proposed item estimation method with prior linear counting method

$$P_{nc} = \left(1 - \frac{1}{m}\right)^{k(\frac{N}{T}-1)} \approx \left(1 - \frac{1}{m}\right)^{k\frac{N}{T}} \tag{3}$$

If the given item e appears in t out of the T measurement periods, then the expected number of cells in the group of item e during the identification phase will be $t \times P_{nc}$ (note: for now, we have ignored the mingling due to hash-print collisions; we will consider that shortly in Section IV-A2). Consequently, the number of encoded bits available in these cells for PIE to use to recover the ID of item e is equal to $r \times t \times P_{nc}$. Using the expression for the decoding failure probability of Raptor codes from Equation (1), the hash-mapping collision survival probability P_m of PIE to recover IDs with length l is calculated by the following equation.

$$P_m = 1 - P_{df}(r \times t \times P_{nc}; l) \tag{4}$$

To calculate the value of P_m, we need the value N of the total number of occurrences of all items in Equation (3), which is not directly known. Next, we present our maximum likelihood based estimation scheme to calculate the estimate $\tilde{N}$ of N.

Let Z_{i0}, Z_{i1}, and Z_{iC} represent the number of empty, singleton, and collided cells, respectively, in the STBF of measurement period i containing m cells. The likelihood that the number of distinct items seen during the measurement period i and stored in the corresponding STBF is n_i is given by the following likelihood function.

$$L(n_i) = \left((1 - \frac{1}{m})^{kn_i}\right)^{Z_{i0}} \times \left(kn_i\frac{1}{m}(1 - \frac{1}{m})^{kn_i-1}\right)^{Z_{i1}} \times \left(1 - (1 - \frac{1}{m})^{kn_i} - kn_i\frac{1}{m}(1 - \frac{1}{m})^{kn_i-1}\right)^{Z_{iC}}$$

To simplify the equation above, we define $q = (1 - \frac{1}{m})^{kn_i}$ and substitute it into this equation. We also substitute $Z_{iC} = m - Z_{i0} - Z_{i1}$ into this equation.

$$L(q) = q^{Z_{i0}} \times \left(\frac{q \ln\{q\}}{(m-1)(\ln\{1 - \frac{1}{m}\})}\right)^{Z_{i1}} \times \left(1 - q - \frac{q \ln\{q\}}{(m-1)(\ln\{1 - \frac{1}{m}\})}\right)^{m - Z_{i0} - Z_{i1}}$$

Taking the first-order derivative of the equation above and equating to 0, we get the sufficient condition that maximizes $L(q)$ as

$$Z_{i1} - qZ_{i1} + (Z_{i0} + Z_{i1})\ln\{q\} + q\{cZ_{i0} - (c+1)m\}\ln\{q\} - cmq\ln\{q^2\} = 0$$

where $c = \frac{1}{m-1} \times \frac{1}{\ln(1-1/m)}$. We numerically calculate the value $\tilde{q}$ of q that satisfies the equation above, and is, thus, the maximum likelihood estimate of q. As $q = (1 - \frac{1}{m})^{kn_i}$, the maximum likelihood estimate $\tilde{n}_i$ of the number of distinct items seen during the measurement period i is given by the following equation.

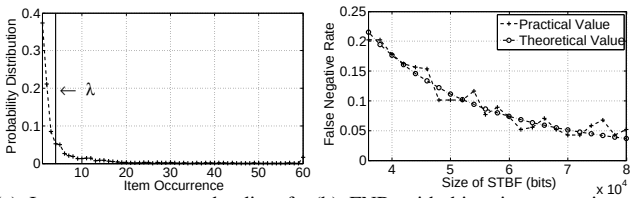
$$\tilde{n}_i = \frac{\ln\{\tilde{q}\}}{k\ln\{1 - \frac{1}{m}\}} \tag{5}$$

As $\tilde{n}_i$ is the estimate of the number of distinct items seen during the measurement period i, the estimate $\tilde{N}$ of the total number of occurrences N during T measurement periods is given by $\tilde{N} = \sum_{i=1}^{T} \tilde{n}_i$.

To show the accuracy of the estimation method proposed above, we compare it with the traditional linear counting method proposed in [37]. The results from these experiments are shown in Figure 3 when the number of item varies between 100 to 13000 given that the number of cells in STBF is $m = 4000$. We observe from this figure that the estimation accuracy of our method is always superior to the estimation accuracy of the linear counting method. The reason behind the superior accuracy of our method is that the linear counting method only uses the information about the number of empty cells and non-empty cells to obtain the estimate, while our method leverages additional information by differentiating between occupied cells and collided cells.

2) Hash-print Collision Survival Probability: Calculating the hash-print collision survival probability P_p is very challenging because we need to calculate the occurrence probability for a given number of cells in a mingled group. For this, we need to enumerate all possible combinations that lead to the same number of cells in a mingled group. For examples, if we have 90 cells in a mingled group, these 90 cells may have resulted from two items with 45 occurrences each, or from three items with 30 occurrences each, *etc.* Thus, this becomes a mathematically and computationally intractable combinatorics problem. To solve the problem of intractability, we present a novel bipartite approximation scheme to calculate P_p. For this, let λ represent a threshold such that if an item $e^{\geq\lambda}$ with occurrences larger than or equal to λ in the T measurement periods makes a group mingled, then no item ID, except that for $e^{\geq\lambda}$, can be recovered from that mingled group. Next, we first calculate hash-print collision survival probability when at least one of the items in a mingled group has $\geq \lambda$ occurrences. We represent this probability with $P_p^{\geq\lambda}$. After this, we calculate hash-print collision survival probability when all of the items in a mingled group have $< \lambda$ occurrences. We represent this probability with $P_p^{<\lambda}$.

To calculate $P_p^{\geq\lambda}$, we first calculate the probability that an item $e^{\geq\lambda}$ with occurrences $\geq \lambda$ makes the group of any given item e mingled, and then calculate the expected value of the number of items with occurrences $\geq \lambda$. The item $e^{\geq\lambda}$ makes the group of the item e mingled when the item $e^{\geq\lambda}$ maps to a specific cell out of the m cells in a given STBF and has the same hash-print as the item e. Considering only a single mapping hash function, this happens with probability $1/m \times 1/2^p$, where p is the number of bits in the hash-print field of each cell. As there are k mapping hash functions, the probability that the item $e^{\geq\lambda}$ does not make the group of the item e mingled is $(1 - 1/m \times 1/2^p)^k$. Let w_t represent the



(a) Item occurrence prob. dist. for DC trace [23] (b) FNR with bipartite approximation

Fig. 4. Dist. of item occurrences and the comparison of theoretical and empirical FNR

percentage of items that appear in t out of T measurement periods. The expected number of occurrences of items, *i.e.*, the average number of measurement periods during which an item appears, is given by $\sum_{t=1}^{T}(w_t \cdot t)$. Thus, the expected number of distinct items is $N/\sum_{t=1}^{T}(w_t \cdot t)$. Consequently, the expected number of items with occurrences $\geq \lambda$ is $N \times \sum_{t=\lambda}^{T} w_t / \sum_{t=1}^{T}(w_t \cdot t)$. As each item is mapped to k cells, the probability that none of the items with occurrences $\geq \lambda$ make the group of the given item e mingled is given by the following equation.

$$P_p^{\geq\lambda} = \left(1 - \frac{1}{m} \times \frac{1}{2^p}\right)^{kN \frac{\sum_{t=\lambda}^{T} w_t}{\sum_{t=1}^{T}(w_t \cdot t)}} \tag{6}$$

To calculate $P_p^{<\lambda}$, we treat each item with $t < \lambda$ occurrences as t distinct items with a single occurrence each. With this notion, the expected number of items with occurrences $< \lambda$ is $N \times \sum_{t=1}^{\lambda-1}(w_t \cdot t)/\sum_{t=1}^{T}(w_t \cdot t)$. Let X_m be a random variable representing the number of such items mapped to a given cell. With k mapping hash functions, it is straightforward to see that X_m follows the binomial distribution $X_m \sim \text{Binom}(k \times N \times \sum_{t=1}^{\lambda-1}(w_t \cdot t)/\sum_{t=1}^{T}(w_t \cdot t), 1/m \times 1/2^p)$. PIE fails to recover the ID of an item e from the mingled group only when the number of all cells from other items $e^<$ with occurrences $< \lambda$ in that group is less than the threshold g_T. Thus, $P_p^{<\lambda}$ is given by the following equation.

$$P_p^{<\lambda} = \sum_{j=0}^{g_T} \binom{j}{kN \frac{\sum_{t=1}^{\lambda-1}(w_t \cdot t)}{\sum_{t=1}^{T}(w_t \cdot t)}} \times \left(\frac{1}{m} \times \frac{1}{2^p}\right)^j \times \left(1 - \frac{1}{m} \times \frac{1}{2^p}\right)^{kN \frac{\sum_{t=1}^{\lambda-1}(w_t \cdot t)}{\sum_{t=1}^{T}(w_t \cdot t)} - j} \tag{7}$$

The overall hash-print collision survival probability P_p is simply the product of $P_p^{\geq\lambda}$ and $P_p^{<\lambda}$, *i.e.*,

$$P_p = P_p^{\geq\lambda} \times P_p^{<\lambda} \tag{8}$$

Discussion: Setting an appropriate value for λ is a critical issue to calculate P_p using our bipartite approximation scheme. An appropriate way to set λ is to put it equal to g_T/P_{nc} because the expected number of cells in a mingled group by an item with occurrences $\geq \lambda$ is $\lambda \times P_{nc}$, and this number must not be greater than the threshold g_T for PIE to be able to successfully decode the IDs of other items in the mingled group. The value of λ can also be selected after performing an empirical study with different values of λ in the range $[2\ g_T]$. The rationale behind developing a bipartite approximation scheme lies in the skewed distribution of item occurrence as well as the small value for g_T. The distributions of item occurrences in practical applications typically follows Zipf or "Zipf-like" distribution, which are highly skewed [38]. We observed this trend in the traces used for evaluation of PIE as well as shown in Figure 4(a). We observe from this figure that almost 75% of all occurrences are contributed by items with individual occurrences no more than 4. Similarly, about 40% of all occurrences are by item with individual occurrence of just 1. As typically λ is proportional to g_T (*e.g.*, $\lambda = g_T/P_{nc}$), and g_T is a small number, λ is also small. Consequently, such a value of λ does not introduce much error when we treat an item with $t < \lambda$ occurrences as t items with a single occurrence each. Thus, the overall error in the calculation of P_p due to the bipartite approximation is very small.

3) False Negative Rate Estimation: We have the following theorem to bound the false negative rate of PIE.

Theorem IV.1. *The false negative rate of PIE is given by*

$$P_{FN} = \left(1 - \frac{\sum_{t=T_{th}}^{T} w_t \times P_m}{\sum_{t=T_{th}}^{T} w_t} \times P_p\right)^k, \tag{9}$$

where w_t is the percentage of items that appear in t out of T measurement periods, P_m is the the hash-mapping collision survival probability, and P_p is the overall hash-print collision survival probability.

To see the accuracy of our proposed calculation of FNR, we conducted a simulation study and compared the FNR obtained through this study with the FNR calculated theoretically using Equation (9). Figure 4(b) plots the FNR calculated from the simulation study as well as the FNR calculated theoretically using Equation (9). For this simulation study, we used the following parameters: N=59752, T=60, T_{th}=40, l=64, k=3, g_T=2, r=3 and p=1. We observe from this figure that the empirically calculated values of FNR lie very close to the theoretically calculated values of FNR. This supports our intuition from the discussion in Section IV-A2 that the overall error in the calculation of P_p due to the bipartite approximation is very small.

B. False Positive Rate

Recall that PIE verifies each ID it recovers by first comparing the hash-print of the recovered ID with the hash-prints stored in the cells of the group and then comparing the hash-print with the hash prints in the cells at locations $h_1(ID), \cdots, h_k(ID)$. False positive rate (FPR) refers to the ratio of non-persistent items, which pass the two verification tests, to the number of all non-persistent items. The following theorem indicates the upper bound of the FPR of PIE.

Theorem IV.2. *The false negative rate of PIE is upper bounded by*

$$P_{FP} = \max\{2^{-\{p+2(r+p)(k-1)P_{nc}\}}, 2^{-\{r+(r+p)(k-1)P_{nc}\}}\} \tag{10}$$

where p is the hash-print length, r is the Raptor code length, k is the number of hash functions, and P_{nc} is the probability that a cell is not collided.

Please see Appendix for the proofs of Theorems IV.1 and IV.2.

V. Parameter Optimization for Persistent Item Identification

In this section, we calculate the optimal values of three parameters, the number of bits r in the Raptor-code field of each cell, the number of bits p in the hash-print field of each cell, and the number of cells m in each STBF. The optimal values of r, p, and m will minimize the FNR of PIE. Next, we will first formulate the optimization problem. After that, we will discuss how to solve this optimization problem to obtain the optimal values of r, p, and m.

A. Optimization Problem Formulation

Let M represent the size of the entire STBF in bits. The value of M can be provided by the system manager, who wants to allocate no more than M bits of SRAM for recording information about the IDs of items. As the size of each cell in bits is $r+p+1$ and there are m cells, the sum of the sizes of all these cells should at most be equal to M. Thus, the formulation of the optimization problem is as follows.

Formulation 1: Minimize P_{FN} such that $m(r+p+1) = M$ and $r, m \in \mathbb{Z}^+, p \in \mathbb{Z}_0^+$.

From Equation (9), we observe that minimizing P_{FN} is equivalent to maximizing P_{sr} as long as the value of k, *i.e.*, the number of hash functions used to map an item to cells, is fixed. The expression for P_{sr} in Equation (21) can be expanded and written as below.

$$P_{sr} = \left(\frac{\sum_{t=T_{th}}^{T} w_t \times (1 - P_{df}(r \times t \times P_{nc}; l))}{\sum_{i=T_{th}}^{T} w_i} \right) \times \left(P_p^{\geq \lambda} \times P_p^{<\lambda} \right)$$

The first term on the right hand side of the equation above depends on $P_{df}(r \times t \times P_{nc}; l)$. When $r \times t \times P_{nc} < l$, the value of $P_{df}(r \times t \times P_{nc}; l)$ is equal to 1. Thus, this first term equals 0. When $r \times t \times P_{nc} > l$, the value of $P_{df}(r \times t \times P_{nc}; l)$ exponentially approaches 0 and this first term almost achieves its maximum possible value of 1. Thus, to maximize P_{sr}, we should ensure that $r \times T_{th} \times P_{nc} \geq \kappa l$, where κ is a preset constant such as $\kappa = 1.01$ or $\kappa = 1.1$. This serves as a new constraint. Our optimization problem can be reformulated as:

Formulation 2: Maximize $(P_p^{\geq \lambda} \times P_p^{<\lambda})$ such that $m(r+p+1) = M$; $r \times T_{th} \times P_{nc} \geq \kappa l$; and $r, m \in \mathbb{Z}^+, p \in \mathbb{Z}_0^+$.

Recall that $P_p^{\geq \lambda}$ and $P_p^{<\lambda}$ are calculated using Equations (6) and (7), respectively. Due to the complexity of these two equations, it is quite challenging to maximize their product. To address this challenge, we take a new approach to find a simpler expression for describing the hash-print collision survival probability P_p that is feasible for parameter optimization. Let's consider the average number of cells corresponding to different IDs but with the same hash-print at a specific cell line. The probability that an item maps to a non-collided cell in this cell line and has the same hash-print as stored in some other cell in this cell-line is $1/2^p \times 1/m \times P_{nc}$. As the total number of occurrences of all items in the T STBFs is N, the average number of mingling cells is $N \times 1/2^p \times 1/m \times P_{nc}$ Intuitively, the probability that the number of mingling cells in a group is less than the threshold g_T, *i.e.*, the hash-print collision survival probability $P_p^{\geq \lambda}$ and $P_p^{<\lambda}$, should be negatively correlated to the average number of mingling cells, which means maximizing the former is equivalent to minimizing the latter. As each item is mapped to k positions, our optimization problem can be reformulated as follows.

Formulation 3: Minimize $(N \times 1/2^p \times 1/m \times P_{nc})^k$ such that $m(r+p+1) = M$; $r \times T_{th} \times P_{nc} \geq \kappa l$; and $r, m \in \mathbb{Z}^+, p \in \mathbb{Z}_0^+$.

Note that in this formulation, we do not see the term w_t, *i.e.*, the probabilistic distribution of occurrences of items. This is a highly desirable property of this formulation because now in implementing our proposed scheme in real applications we no longer need to know such probabilistic distribution a priori.

B. Calculating Optimal Values

Lemma V.1. *Given a fixed value of m and $m(r+p+1) = M$ the false negative rate is minimized when the inequality $r \times T_{th} \times P_{nc} \geq \kappa l$ takes the* equal *sign.*

Lemma V.2. *Given a fixed value r of the size of Raptor code field in cells and $m(r+p+1) = M$, the false negative rate is minimized when the inequality $r \times T_{th} \times P_{nc} \geq \kappa l$ takes the* equal *sign.*

Lemma V.3. *Given a fixed value of p of the hash-print field in cells and $m(r+p+1) = M$, the false negative rate is minimized when the inequality $r \times T_{th} \times P_{nc} \geq \kappa l$ takes the equal sign.*

Combining Lemmas V.1, V.2 and V.3, we get the following following theorem.

Theorem V.1. *The false negative rate is minimized if the inequality $r \times T_{th} \times P_{nc} \geq \kappa l$ takes the equal sign when $m(r+p+1) = M$, and the number of cells m is equal to $\lfloor m^* \rfloor$, where m^* satisfies the following equation*

$$\left(1 - \left(1 + \frac{kN}{T}\right)\frac{1}{m^*}\right) - \ln 2 \cdot \frac{1}{m^*}\left(1 - \frac{1}{m^*}\right) \times \left(M - \frac{\kappa l}{T_{th}}\frac{kN}{T}\left(1 - \frac{1}{m^*}\right)^{-\frac{kN}{T}-1}\right) = 0 \tag{11}$$

and parameters r and p are calculated as below

$$r = \left\lceil \frac{\kappa l}{T_{th}} \left(1 - \frac{1}{m}\right)^{-\frac{kN}{T}} \right\rceil \tag{12}$$

$$p = \left\lfloor \frac{M}{m} \right\rfloor - \left\lceil \frac{\kappa l}{T_{th}} \left(1 - \frac{1}{m}\right)^{-kN/T} \right\rceil - 1 \tag{13}$$

Please see Appendix for the proofs to Lemma V.1, V.2 and V.3, and Theorem V.1.

Discussion The parameter optimization scheme proposed in Theorem V.1 has its inherent limitation. There exists a lower threshold for the objective function in Formulation 3, which can be written in the form of $(\overline{\kappa} \times g_T)^k$ where $\overline{\kappa}$ is a constant that can be empirically estimated. When the value of the objective function in the Formulation 3 falls below $\overline{\kappa} g_T$, the hash-print collision survival probability is very close to 1. Therefore, further minimization at this point is not just meaningless, but could also hurt the overall optimality in some cases. To avoid this problem, we can simply enumerate all possible values of the length of hash-print p from 0 to l to

compute m and r using the equations $m(r+p+1) = M$ and $r \times T_{th} \times P_{nc} = \kappa l$ and then substitute them into Equation (9) to calculate the false negative ratio. Finally, we pick the value of p that leads to the minimum false negative ratio and use the corresponding values of m and r. Overall, such process will terminate within $O(l)$ time, which is fairly fast as l is typically anywhere from 16 bits to 64 bits.

VI. Persistent Item Estimation

Until this point, our discussion has focussed on how PIE identifies persistent items, *i.e.*, how it obtains the IDs of persistent items. Next, we describe how it estimates the number of occurrences of any given persistent item that it has identified. The method that PIE uses can also be used to estimate the number of occurrences of non-persistent items, provided their IDs are already known. Note that PIE only generates the IDs of persistent items, and not of non-persistent items. PIE uses the same two phased approach in estimating the number of occurrence of persistent items that it uses in identifying persistent items. Next, we describe the two phases of PIE from the perspective of estimating the number of occurrence of persistent items.

A. Recording Phase

PIE does not require a separate recording phase. Instead, it uses the information recorded during the recording phase described in Section III-B to estimate the occurrences in addition to identifying the persistent items.

Let X be the random variable representing the number of cells not collided for a given item in a cell line, Y be the random variable representing the number of mingled cells, and Z be the random variable representing the overall observed number of cells. Recall from Section IV-A1 that X follows a binomial distribution, *i.e.*, $X \sim Binom(\mathcal{N}_i, \mathcal{P}_{cs})$ where $\mathcal{P}_{cs} = P_{nc} = (1 - \frac{1}{m})^{kN/T}$. Note that $\mathcal{N}_i$ represents the number of occurrences of the i-th persistent item. Furthermore, a cell mingles with another cell if and only if both of their stored items share the same Raptor code and fingerprint, which happens with probability $\mathcal{P}_{ms} = \frac{1}{m}\frac{1}{2^{r+p}}$. As the expected number of times an item is recorded during a measurement period is $\frac{kN}{T}$, we get $Y \sim Binom(\frac{kN}{T}, \mathcal{P}_{ms})$. The random variables X and Y are approximately independent of each other. Theorem VI.1 gives the expressions for probability distribution, expected value, and variance of X.

Theorem VI.1. *Let Z be the random variable representing the overall observed number of cells. The probability distribution, expected value, and variance of Z are given by the following equations.*

$$P\{Z = z\} = \sum_{z_f=0}^{z} \left\{ \binom{\mathcal{N}_i}{z_f} (\mathcal{P}_{cs})^{z_f} (1-\mathcal{P}_{cs})^{\mathcal{N}_i - z_f} \times \binom{kN/T}{z - z_f} (\mathcal{P}_{ms})^{z-z_f} (1-\mathcal{P}_{ms})^{kN/T - z + z_f} \right\} \tag{14}$$

$$E(Z) = \mathcal{N}_i \cdot \mathcal{P}_{cs} + \frac{kN}{T} \cdot \mathcal{P}_{ms} \tag{15}$$

$$Var(Z) = \mathcal{N}_i \cdot \mathcal{P}_{cs} \cdot (1-\mathcal{P}_{cs}) + \frac{kN}{T} \cdot \mathcal{P}_{ms} \cdot (1-\mathcal{P}_{ms}) \tag{16}$$

Proof. Please see Appendix for the proof. □

B. Querying Phase

In the querying phase, our objective is to calculate $\tilde{\mathcal{N}}_i$, the estimate of the number of occurrences of a given item based on the observed number of cells Z. Theorem VI.2 gives the expression to calculate $\tilde{\mathcal{N}}_i$ using the maximum likelihood estimation (MLE) method.

Theorem VI.2. *Let $\tilde{Z}[j]$ $(j = 1, \ldots, k)$ denote the observed numbers of cells in k mapped cell lines of a given item. The estimate $\tilde{\mathcal{N}}_i$ of the number of occurrences $\mathcal{N}_i$ of a given item is given by the numerical solution of the following equation*

$$\sum_{j=1}^{k} \left\{ \frac{\sum_{z_f=0}^{\tilde{Z}[j]} \{\mathbf{F}(j) \times \mathbf{P}(j)\}}{\sum_{z_f=0}^{\tilde{Z}[j]} \{\mathbf{F}(j) \times \mathbf{Q}(j)\}} \right\} = 0 \tag{17}$$

where

$$\mathbf{F}(j) = \binom{\frac{kN}{T}}{\tilde{Z}[j] - z_f} (\mathcal{P}_{ms})^{\tilde{Z}[j]-z_f} (1-\mathcal{P}_{ms})^{\frac{kN}{T} - \tilde{Z}[j] + z_f}$$

$$\mathbf{P}(j) = \binom{\mathcal{N}_i}{z_f} (\mathcal{P}_{cs})^{z_f} (1-\mathcal{P}_{cs})^{\mathcal{N}_i - z_f} \left(\psi^{(0)}\{\mathcal{N}_i + 1\} - \psi^{(0)}\{\mathcal{N}_i + 1 - z_f\} + \ln(1-\mathcal{P}_{cs}) \right)$$

$$\mathbf{Q}(j) = \binom{\mathcal{N}_i}{z_f} (\mathcal{P}_{cs})^{z_f} (1-\mathcal{P}_{cs})^{\mathcal{N}_i - z_f}$$

where $\psi^{(0)}\{.\}$ is the 0^{th} order polygamma function.

Proof. Please see Appendix for the proof. □

C. Accuracy Analysis

Next, we analyze the estimation accuracy of our MLE based estimation method. Formally, we derive expressions to estimate parameters of MLE for a given confidence interval β and given required success probability α such that the estimated value $\tilde{\mathcal{N}}_i$ satisfies the requirement $P\{|\tilde{\mathcal{N}}_i - \mathcal{N}_i| \leq \beta \mathcal{N}_i\} \geq \alpha$. For this, we use the classical theoretical result from MLE theory that when the number of observations is sufficiently large, the distribution of estimation error approaches zero-mean normal distribution, *i.e.*, $\sqrt{k}(\tilde{\mathcal{N}}_i - \mathcal{N}_i) \rightarrow Norm(0, \frac{1}{\mathcal{I}(\tilde{\mathcal{N}}_i)})$, or equivalently

$$(\tilde{\mathcal{N}}_i - \mathcal{N}_i) \rightarrow Norm\left(0, \frac{1}{k\mathcal{I}(\tilde{\mathcal{N}}_i)}\right) \tag{18}$$

where $\mathcal{I}(\tilde{\mathcal{N}}_i)$ denotes fisher information of L, defined as

$$\mathcal{I}(\tilde{\mathcal{N}}_i) = -E\left[\frac{\partial^2}{\partial \mathcal{N}_i^2} \ln\{L\}\right] \tag{19}$$

As the expression of L in Equation (29) is quite complicated, we apply the following approximation to make it tractable. First, we approximate the binomial distributions of X and Y as two normal distributions $X \sim Norm(\mathcal{N}_i\mathcal{P}_{cs}, \mathcal{N}_i\mathcal{P}_{cs}(1-\mathcal{P}_{cs}))$ and $X \sim Norm(\frac{kN}{T}\mathcal{P}_{ms}, \frac{kN}{T}\mathcal{P}_{ms}(1-\mathcal{P}_{ms}))$, respectively. Then, the approximated distribution of Z can be written as $Norm(\mathcal{N}_i\mathcal{P}_{cs} + \frac{kN}{T}\mathcal{P}_{ms}, \mathcal{N}_i\mathcal{P}_{cs}(1-\mathcal{P}_{cs}) + \frac{kN}{T}\mathcal{P}_{ms}(1-\mathcal{P}_{ms})) = Norm(\mu, \sigma^2)$ as $Z = X + Y$. As a result, we get

$$\frac{\partial^2 \ln\{L\}}{\partial \mathcal{N}_i^2} = \frac{\partial^2}{\partial \mathcal{N}_i^2} \ln\left\{ \prod_{j=1}^{k} \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Z[j]-\mu)^2}{2\sigma^2}} \right) \right\}$$

$$= \sum_{j=1}^{k} \frac{\partial^2}{\partial \mathcal{N}_i^2} \left(-\ln\sqrt{2\pi\sigma^2} - \frac{(Z[j]-\mu)^2}{2\sigma^2} \right)$$

$$= \sum_{j=1}^{k} \left[-\frac{\mu'^2}{\sigma^2} + \frac{((\sigma^2)' + 4(\mu - Z[j])\mu')(\sigma^2)'}{2(\sigma^2)^2} - \frac{(\mu - Z[j])^2((\sigma^2)')^2}{(\sigma^2)^3} \right]$$

As $E(\mu - Z[j]) = 0$, $E[(\mu - Z[j])^2] = \sigma^2$, $\mu' = \mathcal{P}_{cs}$, and $(\sigma^2)' = \mathcal{P}_{cs}(1 - \mathcal{P}_{cs})$, we get

$$\begin{aligned}
\mathcal{I}(\tilde{\mathcal{N}_i}) &= -E\left[\frac{\partial^2}{\partial \mathcal{N}_i^2} \ln\{L\}\right] \\
&= -\sum_{j=1}^{k} E\left[-\frac{\mu'^2}{\sigma^2} + \frac{((\sigma^2)' + 4(\mu - Z[j])\mu')(\sigma^2)'}{2(\sigma^2)^2} - \frac{(\mu - Z[j])^2((\sigma^2)')^2}{(\sigma^2)^3} \right] \\
&= -\sum_{j=1}^{k} \left[-\frac{\mu'^2}{\sigma^2} - \frac{((\sigma^2)')^2}{2(\sigma^2)^2} \right] \\
&= k\left[\frac{\mu'^2}{\sigma^2} + \frac{((\sigma^2)')^2}{2(\sigma^2)^2}\right] \\
&= k\left[\frac{2\sigma^2\mu'^2 + ((\sigma^2)')^2}{2(\sigma^2)^2}\right] \\
&= \frac{k\mathcal{P}_{cs}^2[2\sigma^2 + (1 - \mathcal{P}_{cs})^2]}{2(\sigma^2)^2}
\end{aligned}$$

As per Equation (18), the variance of $\tilde{\mathcal{N}_i}$ is $Var(\tilde{\mathcal{N}_i}) = \frac{1}{k\mathcal{I}(\tilde{\mathcal{N}_i})}$. Thus, for a given required success probability α along with the values of remaining parameters, where Z_α is the α percentile for the standard Gaussian distribution, and $\sigma^2 = \mathcal{N}_i\mathcal{P}_{cs}(1 - \mathcal{P}_{cs}) + \frac{kN}{T}\mathcal{P}_{ms}(1 - \mathcal{P}_{ms})$, the confidence interval is thus given by the following equation.

$$\beta = Z_\alpha \sqrt{\frac{2(\sigma^2)^2}{k^2\mathcal{P}_{cs}^2[2\sigma^2 + (1 - \mathcal{P}_{cs})^2]}} \tag{20}$$

This equation can be used to estimate the values of unknown parameters for given values of α and β.

VII. Performance Evaluation

We implemented and extensively evaluated PIE along with three other schemes namely Invertible Bloom Filter (IBF) [19], [20], Count-Min (CM) sketch [7], and Small-Space [15], [16] in Matlab. Next, we first describe the three network traces that we used for evaluation. After that, we present our results from the evaluation of PIE's FNR and FPR and compare them with IBF, CM sketch, and Small-Space. Then, we study the performance of PIE by optimizing coding efficiency and memory access efficiency. Last, we show the occurrence estimation accuracy of PIE.

A. Item Traces and Setup

Item Traces: To evaluate the performance of PIE, we use three real network traces CHIC [21], ICSI [22], and DC [23]. CHIC is a backbone header trace, published by CAIDA and collected in 2015, which includes the arrival times of packets at a 10GigE link interface along with the flow IDs associated with those packets. For this evaluation, we used HTTP flows from 6 minutes of packet capture. We treat each packet as an item and the 5-tuple flow ID of each packet as the item ID. ICSI is an enterprise network traffic trace collected at a medium-sized enterprise network. From this trace, we use TCP traces generated from 22 different ports in one hour of packet capture. DC is a data center traffic trace collected at a university data center collected for a little more than an hour. From this trace, we use TCP traces generated from one hour of packet capture. Table II reports the total duration, number of packets and number of distinct flows of these three network traces. Figures 5 and 6 show the item occurrence probability distributions and item persistence probability distributions for the three network traces, respectively. We can see from Figure 6 that the persistence values of items are highly skewed, and their probability distributions roughly follow Zipf distribution, except for some abnormal points. Note that we move Table II and all the figures for evaluation to Appendix to save space.

System Parameters: In our experiments, for each trace, we set the number of measurement periods as $T = 60$, the constant for Raptor code decoding $\kappa = 1.05$, the number of mapping hash functions $k = 3$, the mingling threshold $g_T = 1$, and the amount of memory allocated to build STBF $M = 600$kbits for CHIC, $M = 100$kbits for ICSI, and $M = 300$kbits for DC, respectively.

Baseline Setup: Small-Space can be directly applied to address our problem. Particularly, Small-Space uses a sketch to record ID, index of the current measurement period, and frequency value of sampled items. It samples items using a hash function that is based on the item ID and the measurement period in which it arrived in. Small-Space stores each ID at its original size, which is up to 16 bytes for all datasets. We cannot use hashes of IDs because Small-Space needs the whole ID information for sampling. Unlike Small-Space, IBF and CM sketch cannot be directly applied to identify persistent items; therefore, we adapt them to make performance comparison possible. For IBF, instead of employing a single IBF throughout all measurement periods, we allocate one IBF to each measurement period. As IBF cannot tell whether an incoming item is recorded or not, we need an additional conventional Bloom filter (BF) to store existence information of items. When an arriving item is not bound in the associated conventional BF, we add its information to both conventional BF and IBF. For CM sketch, we again need a conventional BF to avoid counting of multiple repetitions of the same item in any given measurement period. Furthermore, for fair comparison, we allocate the same amount of memory to build IBF or CM sketch as that to build PIE. Moreover, to the advantage of IBF and CM sketch, we neglect the storage cost of these assisting conventional BFs for both IBF and CM sketch. We also ignore the space cost of CM sketch for recording IDs of persistent items, which would be considerable when the number of distinct items is large.

B. False Negative Rate

We first present the FNR of PIE. For this, we set a maximum desired value of FNR, calculate the optimal values of r, p, and m for the maximum desired FNR as described in Section V-B, perform the simulations, and compare the FNR obtained from the simulations with the maximum desired FNR. We repeat this process for multiple values of maximum desired FNR. After this, we compare the FNR of PIE with the FNR of IBF. We do not compare the FNR of PIE with the FNR of CM sketch because the FNR for CM sketch is always 0.

Our results show that the average FNR of PIE calculated from simulations is always less than the maximum desired FNR. We vary the maximum desired FNR from 0.01 to 0.1 and for each value of FNR, calculate the parameters r, p, and m for the three traces. For each set of parameters, we randomly select part of the data with fixed size from each of three networks traces, and repeat the experiments 50 times. This gives us 50 values of FNR from simulations for each value of maximum desired FNR. We do all these experiments with two values for the threshold T_{th}, *i.e.*, $T_{th} = 40$ and $T_{th} = 50$; and plot the results in Figures 7 and 8, respectively. We plot the 50 values of FNR obtained from simulations for each value of maximum desired FNR as a box-plot. On each box-plot, the central mark is the median, the edges of the box are the 25-th and 75-th percentiles, and the crosses are the outliers. The dashed lines in each figure corresponds to the function $y = x$. From these figures, we observe that the average values of FNR obtained from simulations are always less than the maximum desired FNR. This indicates that PIE can be configured to achieve any desired value of FNR by appropriately adjusting the parameters r, p, and m.

Our results show that the average FNR of PIE is almost twice an order of magnitude smaller than the FNR of IBF, and is 7.7% smaller than the FNR of Small-Space. Figures 9 and 10 plot the FNRs of PIE, IBF and Small-Space for $T_{th} = 40$ and $T_{th} = 50$, respectively. These figures also plot the FNR of PIE calculated theoretically using Equation (9). We observe that the theoretically calculated FNR closely matches the empirically calculated FNR of PIE. We also observe from these figures that FNRs of PIE, IBF and Small-Space decrease with the increase in the number of bits M allocated to their corresponding data structures. From our experiments, we calculated that on average, PIE achieves 19.5 times smaller FNR compared to IBF. We observe from these figures that the FNR of IBF remains almost 1 when M is small, and then drops rapidly as the value of M increases. The reason behind this observation is that the recovering process of IBF resembles the "peeling process" of finding a 2-core in random hypergraphs, and its success probability of recovering rockets when the percentage of cells that store only one item, which is determined by M, is sufficiently large. Note that IBF achieves a better performance compared to PIE only when the size of its data structure is significantly large. For example, IBF needs 1.12×10^5 bits of storage space for DC trace, which is so large that it equals half the space required to store all occurrences of items along with their original IDs. This cost may not be acceptable for real applications. Besides, in general the false positives of the DC and ICSI traces for $T_{th} = 40$ are larger than that for $T_{th} = 50$, while the false positive of the CHIC trace is the opposite. This is because that more occurrences of items will give rise to not only larger number of items surviving collision which is favorable for decoding, but also larger number of mingled items which is unfavorable for decoding, which makes the impact of T_{th} complicated.

C. False Positive Rate

For FPR, we compare the FPR of PIE with the FPR of CM sketch. We do not compare with IBF because the FPR for IBF is always 0. For the simulations to obtain results for this section, we fix the length of Raptor code field in cells of STBF to 2, and set the size of each counter in CM sketch to 8 bits, which is sufficient to record items that appear in all 60 measurement periods.

Our results show that FPR of PIE is at least 426.1 and 1898.2 times less than the FPRs of CM sketch and Small-Space, respectively. Figures 11 and 12 plot the FPR of PIE, CM sketch and Small-Space along with the theoretically bound on FPR calculated using Equation 10 for $T_{th} = 40$ and $T_{th} = 45$, respectively. We observe that the empirically obtained FPR of PIE is always less than the theoretically calculated bound. We also observe from these figures that the FPR of PIE becomes 0 when the length of hash-print field exceeds 5 for $T_{th} = 40$ and when it exceeds 3 for $T_{th} = 45$ for all three traces. Even in the worst case, the FPR of PIE is just 24.9% the FPR of CM sketch, and 28.7% the FPR of Small-Space. These figures further show that the FPRs of both PIE, CM sketch and Small-Space improve with increasing p. This happens because larger p means fewer hash-print collisions and more success during verification steps for PIE and more counters for CM sketch and Small-Space.

We also observe from our experiments that overall, the FPR of CM sketch increase with T_{th} while the FPR of PIE decrease with T_{th}. As described in [7], the estimate occurrence $\hat{i}$ of an item obtained through CM sketch is subject to $\hat{i} \leq i + B \times N$ with some probability α. Though the absolute error of estimation, *i.e.*, $B \times N$, is roughly stable, the number of eligible items decreases with an increasing T_{th}. This gives rise to an increasing FPR for CM-sketch. On the contrary, for PIE, with a large T_{th}, there are potentially more cells recording a given item, which provides more opportunities for verification of item ID and thus reduces the FPR.

D. Coding Efficiency and Memory Access Efficiency

We proceed to evaluate the coding efficiency and memory access efficiency. We only study DC trace. We set $e_{th} = 0$ and $T_{th} = 10$. We set a fixed threshold 70% of false negative for coding efficiency as well as memory access efficiency evaluation. Our goal is then to minimize r or minimize $r + p$ given a space budget for STBF.

Our results show that both coding efficiency and memory access efficiency improve as the size of STBF increases. Figures 13 and 14 show both r and $r + p$ decrease with an increasing STBF size M. It makes intuitive sense because higher resources should lead to higher performance. Nevertheless, the decreasing speed slows down as the size of STBF M increases.

E. Occurrence Estimation Accuracy

Our results show that PIE always achieves the required reliability. Figure 15 shows the CDFs of the observed value of β, which can be computed by $|\tilde{\mathcal{N}} - \mathcal{N}|/\mathcal{N}$, for three network traces CHIC, DC, and ICSI when $\alpha = 0.9$ and $\beta = 0.1$. We can observe that for each curve in this figure, its CDF is larger than its required reliability $\alpha = 0.9$ when its observed β reaches its required confidence interval 0.1. Similarly, Figure 16 demonstrates the CDFs of the observed value of β when $\alpha = 0.95$ and $\beta = 0.05$. All three curves in Figure 16 have their CDF larger than their required reliability $\alpha = 0.95$ when

observed β exceeds 0.05. This shows our scheme PIE always achieves the required reliability.

F. Scalability Performance and Threshold Sensitivity

To study the scalability performance and threshold sensitivity of our algorithm, we conduct experiments on a new larger dataset, named WorldCup98, compared with CHIC, DC, and ICSI. WorldCup98 consists of over 100 million requests and 64 million distinct identifiers from the first 20 days in WORLDCUP98 [39].

Our results show that both FNRs and FPRs of all algorithms degrade as the size of data increases. Figures 17 and 18 respectively show the FNRs and FPRs of all the three algorithms when the applied portion of the data trace increases from 55% to 100%. This makes sense because given that the allocated memory sizes for all the algorithms keep unchanged, the probabilities of hash-mapping collision and hash-print collision will increase, which lead to high FNRs and FPRs.

Our results show that the FNRs increase with the threshold T_{th} while the FPRs decrease with T_{th} for all the three algorithms. Figures 19 and 20 show the FNRs and FPRs for all the three algorithms when T_{th} increases from 70 to 94, respectively. Note that the total number of measurement periods is set to 100. The FNR of PIE generally increases with T_{th} because the decoding requirement becomes more stringent when encoded bits need to be successfully extracted in more measurement periods. Moreover, FPR of PIE steadily decreases with T_{th} because we have more encoded bits for verification and the probability that a false positive passes the verification tests decreases.

VIII. Conclusion

The key contribution of this paper is in proposing a scheme to identify persistent items and estimate their number of occurrences. The key technical novelty of this paper is in proposing the Space-Time Bloom filter, which uses Raptor codes and hash-prints to efficiently store information about the IDs of items in such a way that the IDs of persistent items can be recovered with the desired FNR and FPR, and their occurrences can be estimated with the desired confidence level and success probability. The key technical depth of this paper is in calculating the FNR and FPR and using them to obtain optimal values of system parameters, and in analyzing the accuracy of our occurrence estimation scheme. Our theoretical analysis and experimental results show that PIE always achieves the desired FNR and its FNR is always less than the FNRs of prior schemes adapted for persistent item identification. Our theoretical analysis and experimental results also show that PIE always achieves the required success probability in estimating the number of occurrence of persistent items. PIE is scalable in that it is designed to store information about the IDs of all items in a single data structure, the STBF, in a given measurement period and periodically transfers the contents of the STBF to permanent storage.

Acknowledgment

Alex X. Liu is also affiliated with Michigan State University.

References

[1] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Quicksand: Quick summary and analysis of network data," Tech. Rep.

[2] M. H. S. Bangalore, "Resource adaptive technique for frequent itemset mining in transactional data streams," *IJCSNS*, vol. 12, no. 10, p. 87, 2012.

[3] S. Moro, R. Laureano, and P. Cortez, "Using data mining for bank direct marketing: An application of the crisp-dm methodology," in *Proc. ESM*. Eurosis, 2011, pp. 117–121.

[4] C. C. Aggarwal, "An introduction to sensor data analytics," in *Managing and Mining Sensor Data*. Springer, 2013, pp. 1–8.

[5] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Data stream mining," in *Data Mining and Knowledge Discovery Handbook*. Springer, 2010, pp. 759–787.

[6] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *Proc. VLDB*. VLDB Endowment, 2002, pp. 346–357.

[7] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[8] A. Metwally, D. Agrawal, and A. El Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *Database Theory-ICDT 2005*. Springer, 2005, pp. 398–412.

[9] G. Cormode and M. Hadjieleftheriou, "Finding the frequent items in streams of data," *Communications of the ACM*, vol. 52, no. 10, pp. 97–105, 2009.

[10] H. Liu, Y. Lin, and J. Han, "Methods for mining frequent items in data streams: an overview," *Knowledge and information systems*, vol. 26, no. 1, pp. 1–30, 2011.

[11] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, "Interpreting the data: Parallel analysis with sawzall," *Scientific Programming*, vol. 13, no. 4, pp. 277–298, 2005.

[12] Q. Xiao, Y. Qiao, M. Zhen, and S. Chen, "Estimating the persistent spreads in high-speed networks," in *Proc. ICNP*. IEEE, 2014, pp. 131–142.

[13] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and D. Papagiannaki, "Exploiting temporal persistence to detect covert botnet channels," in *Recent Advances in Intrusion Detection*. Springer, 2009, pp. 326–345.

[14] N. Immorlica, K. Jain, M. Mahdian, and K. Talwar, "Click fraud resistant methods for learning click-through rates," in *Internet and Network Economics*. Springer, 2005, pp. 34–45.

[15] B. Lahiri, J. Chandrashekar, and S. Tirthapura, "Space-efficient tracking of persistent items in a massive data stream," in *Proc. DEBS*. ACM, 2011, pp. 255–266.

[16] B. Lahiri, S. Tirthapura, and J. Chandrashekar, "Space-efficient tracking of persistent items in a massive data stream," *Statistical Analysis and Data Mining*, vol. 7, no. 1, pp. 70–92, 2014.

[17] S. A. Singh and S. Tirthapura, "Monitoring persistent items in the union of distributed streams," *Journal of Parallel and Distributed Computing*, vol. 74, no. 11, pp. 3115 – 3127, 2014.

[18] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[19] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *Proc. Allerton*. IEEE, 2011, pp. 792–799.

[20] D. Eppstein, M. T. Goodrich, F. Uyeda, and G. Varghese, "What's the difference?: efficient set reconciliation without prior context," in *Proc. SIGCOMM*, vol. 41, no. 4. ACM, 2011, pp. 218–229.

[21] "The caida ucsd anonymized 2011 internet traces." http://www.caida.org/data/overview.

[22] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney, "A first look at modern enterprise traffic," in *Proc. IMC*, 2005, pp. 15–28.

[23] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. IMC*. ACM, 2010, pp. 267–280.

[24] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Automata, Languages and Programming*. Springer, 2002, pp. 693–703.

[25] H. Dai, M. Shahzad, A. X. Liu, and Y. Zhong, "Finding persistent items in data streams," *Proc. VLDB*, vol. 10, no. 4, pp. 289–300, 2016.

[26] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[27] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proc. CoNEXT*. ACM, 2014, pp. 75–88.

[28] Y. Li, H. Wu, T. Pan, H. Dai, J. Lu, and B. Liu, "Case: Cache-assisted stretchable estimator for high speed per-flow measurement," in *Proc. INFOCOM*. IEEE, 2016, pp. 1–9.

[29] T. Yang, A. X. Liu, M. Shahzad, D. Yang, Q. Fu, G. Xie, and X. Li, "A shifting framework for set queries," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3116–3131, 2017.

[30] L. Wang, T. Yang, H. Wang, J. Jiang, Z. Cai, B. Cui, and X. Li, "Fine-grained probability counting for cardinality estimation of data streams," *World Wide Web Journal*, pp. 1–17, 2018.

[31] T. Yang, Y. Zhou, H. Jin, S. Chen, and X. Li, "Pyramid sketch: A sketch framework for frequency estimation of data streams," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1442–1453, 2017.

[32] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," in *ACM SIGCOMM. Proceedings of the ACM SIGCOMM Conference*, 2018.

[33] T. Yang, L. Wang, Y. Shen, M. Shahzad, Q. Huang, X. Jiang, K. Tan, and X. Li, "Empowering sketches with machine learning for network measurements," in *ACM SIGCOMM workshop on NetAI*, 2018.

[34] T. Yang, J. Gong, H. Zhang, L. Zou, L. Shi, and X. Li, "Heavyguardian: Separate and guard hot items in data streams," in *ACM SIGKDD*, 2018.

[35] T. Yang, A. X. Liu, M. Shahzad, Y. Zhong, Q. Fu, Z. Li, G. Xie, and X. Li, "A shifting bloom filter framework for set queries," *Proceedings of the VLDB Endowment*, vol. 9, no. 5, pp. 408–419, 2016.

[36] R. M. Transport, "Raptorq forward error correction scheme for object delivery," IETF Internet Draft, Tech. Rep., 2011.

[37] K.-Y. Whang, B. T. Vander-Zanden, and H. M. Taylor, "A linear-time probabilistic counting algorithm for database applications," *ACM Transactions on Database Systems*, vol. 15, pp. 208–229, 1990.

[38] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 219–232, 2004.

[39] "The worldcup98 trace." http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

[40] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Courier Corporation, 1964, no. 55.

Haipeng Dai received the B.S. degree in the Department of Electronic Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2010, and the Ph.D. degree in the Department of Computer Science and Technology in Nanjing University, Nanjing, China, in 2014. His research interests are mainly in the areas of wireless charging, mobile computing, and data mining. He is a research assistant professor in the Department of Computer Science and Technology in Nanjing University. His research papers have been published in many prestigious conferences and journals such as ACM MobiSys, ACM MobiHoc, ACM VLDB, ACM SIGMETRICS, IEEE INFOCOM, IEEE ICDCS, IEEE ICNP, IEEE SECON, IEEE IPSN, IEEE JSAC, IEEE/ACM TON, IEEE TMC, IEEE TPDS, and IEEE TOSN. He is an IEEE and ACM member. He serves/ed as Poster Chair of the IEEE ICNP'14, TPC member of the IEEE ICNP'14, IEEE ICC'14-18, IEEE ICCCN'15-18 and the IEEE Globecom'14-18. He received Best Paper Award from IEEE ICNP'15, Best Paper Award Runner-up from IEEE SECON'18, and Best Paper Award Candidate from IEEE INFOCOM'17.

Muhammad Shahzad received the Ph.D. degree in computer science from Michigan State University in 2015. He is currently an Assistant Professor with the Department of Computer Science, North Carolina State University, USA. His research interests include design, analysis, measurement, and modeling of networking and security systems. He received the 2015 Outstanding Graduate Student Award, the 2015 Fitch Beach Award, and the 2012 Outstanding Student Leader Award at Michigan State University.

Alex X. Liu received his Ph.D. degree in Computer Science from The University of Texas at Austin in 2006. He received the IEEE & IFIP William C. Carter Award in 2004, a National Science Foundation CAREER award in 2009, and the Michigan State University Withrow Distinguished Scholar Award in 2011. He is an Associate Editor of IEEE/ACM Transactions on Networking, an Editor of IEEE Transactions on Dependable and Secure Computing, and an Area Editor of Computer Communications. He received Best Paper Awards from ICNP-2012, SRDS-2012, and LISA-2010. His research interests focus on networking and security.

Meng Li received his B.E. degree in Computer Science from Nanjing University, Jiangsu, China, in 2016. He is currently a Ph.D student in Nanjing University. His research interests are in the area of of data mining.

Yuankun Zhong received his B.E. degree in Computer Science from Nanjing University, Jiangsu, China, in 2010. He is studying for his master degree in the Department of Computer Science and Technology in Nanjing University. His research focuses on networking and security.

Guihai Chen received B.S. degree in computer software from Nanjing University in 1984, M.E. degree in computer applications from Southeast University in 1987, and Ph.D. degree in computer science from the University of Hong Kong in 1997. He is a professor and deputy chair of the Department of Computer Science, Nanjing University, China. He had been invited as a visiting professor by many foreign universities including Kyushu Institute of Technology, Japan in 1998, University of Queensland, Australia in 2000, and Wayne State University, USA during Sept. 2001 to Aug. 2003. He has a wide range of research interests with focus on sensor networks, peer-to-peer computing, high-performance computer architecture and combinatorics.

APPENDIX

A. Notations Used in This Paper

TABLE I
NOTATIONS AND SYMBOLS USED IN THE PAPER

Symbol	Description
T	Number of measurement periods
m	Size of STBF in cells
C_i	Array of cells of STBF in a measurement period i
M	Size of STBF in bits
k	Number of hash functions for STBF
$C_{iR}[x]$	Raptor code field of cell x
$C_{iP}[x]$	Hash-print field of cell x
$C_{iF}[x]$	Flag field of cell x
r	Length of Raptor codes in a cell
p	Length of hash-print in a cell
l	Length of item ID in bits
N	Number of total occurrences of all items
$\mathcal{N}_i$	Number of occurrences of the i-th persistent item
$P_{df}(r;l)$	Decoding failure probability when using r bits of Raptor codes for decoding, where l is the length of item ID in bits
g	Number of cells in a group
g_r	Number of temporarily removed cells from a group for recovering purpose
g_T	Threshold of the number of removed cells during recovering process
P_m	Hash-mapping collision survival probability
P_p	Hash-print collision survival probability
P_{nc}	Probability that a cell in a given STBF to which a given item e maps is not collided
$e^{\geq\lambda}$	An item with occurrences larger than or equal to λ
$P_p^{\geq\lambda}$	Hash-print collision survival probability when at least one of the items in a mingled group has $\geq\lambda$ occurrences
$P_p^{<\lambda}$	Hash-print collision survival probability when all of the items in a mingled group have $<\lambda$ occurrences
λ	Boundary parameter in bipartite approximation scheme
w_t	Percentage of items that appear in t out of T measurement periods
P_{sr}	Expected probability of successful recovery of an item at a specific cell
P_{FN}	False negative rate for persistent item identification
P_{FP}	False positive rate for persistent item identification
κ	Constant established for Raptor code decoding
α	Confidence interval
β	Success probability

B. Proof of Lemma III.1

Proof. By [18], the computational overhead for encoding the ID of l-bit length into r bits using Raptor codes is $O(r \cdot l)$. Moreover, generating p-bit hash-prints needs $O(p)$ time and mapping an item to k locations using k hash functions needs $O(k)$ time. And modifying the stored information requires $O(k\cdot(r+p))$ time. Then, the overall computational overhead is $O(r\cdot l)+O(p)+O(k)+O(k\cdot(r+p)) = O(r\cdot l+k\cdot(r+p))$. □

C. Identification Algorithm of PIE

Algorithm 2: Identification Algorithm of PIE

Input: All T STBFs for all measurement periods
Output: Identified persistent items

Initialize the persistent item set to be empty;
for *each out of m cell positions in the T STBFs* **do**
  Pick all the cells $C_1[x]$, $C_2[x]$, $\cdots$, $C_T[x]$ $(1 \leq x \leq m)$ at the same cell position from the T STBFs to constitute a so-called *cell line*;
  Discard all empty and collided cells in this cell line because such cells do not contain information about any item;
  Divide the remaining cells in the cell line into groups such that the hash-prints of all cells in the same group are the same;
  Compare the hash-prints of the groups generated from the current cell line with the hash-prints of the groups generated from the previous cell lines, and merge the groups with the same hash-prints;
for *each obtained group* **do**
  Regenerate the encoding-coefficient vectors $\mathbf{a}_{ij}$ $(1 \leq j \leq r)$ using the index i of the measurement period;
  Observes the values of R_{ij}^e from the cells in the group, plug in all the values of $\mathbf{a}_{ij}$ and R_{ij}^e into Equation (2) and obtain a set of $g \times r$ linear equations;
  if *$g \times r$ is no less than the length of item ID l* **then**
    Solve $g \times r$ linear equations to get the item ID vector $\mathbf{I}^e$;
    Calculate and compare it with the hash-prints in the cells of the group;
    if *the hash-print of the recovered ID matches the hash-prints in the cells of the group* **then**
      if *the regenerated Raptor code field and hash-print of the recovered ID match the Raptor code fields and hash-prints stored in the cells at locations $h_1(ID), \cdots, h_k(ID)$ in all those STBFs from which the cells in the group were taken* **then**
        Add the recovered item ID to the persistent item set;
    else
      for *g_r from 1 to g_T* **do**
        forall *possible cases by removing g_r cells from the group* **do**
          Try to recover the ID using the remaining $g - g_r$ cells;
          if *the hash-print of the recovered ID matches the hash-prints in the cells of the group **and** the regenerated Raptor code field and hash-print of the recovered ID match the Raptor code fields and hash-prints stored in the cells at locations $h_1(ID), \cdots, h_k(ID)$ in all those STBFs from which the cells in the group were taken* **then**
            Add the recovered item ID to the persistent item set.
return *the persistent item set*

D. Proof of Lemma III.2

Proof. We sketch the proof as follows. Apparently, the dominant part of computational overhead is caused by processing mingled groups. In the worst case, there are $O(m \cdot T)$ groups to be processed, and all groups are mingled groups, which incurs an enumeration including $\binom{kT}{1}+\ldots+\binom{kT}{g_T} = O(\binom{KT}{g_T})$ $(g_T \ll kT)$ rounds. Moreover, solving a system with $O(l)$ linear equations requires $O(l^3)$ time using methods like Gaussian elimination. To sum up, the result follows. □

E. Proof of Theorem IV.1

Proof. An item with t occurrences, where $T_{th} \leq t \leq T$, can be recovered if and only if it survives both hash-mapping collisions as well as hash-print collisions, *i.e.*, the probability of the successful recovery of the ID of an item is $P_m \times P_p$, where P_m and P_p are calculated in Equations (4) and (8), respectively. The expected probability of successful recovery P_{sr} for an item at a specific cell can be derived by taking into account all possible long-duration items as shown in the following equation.

$$P_{sr} = \frac{\sum_{t=T_{th}}^{T} w_t \times P_m}{\sum_{t=T_{th}}^{T} w_t} \times P_p \tag{21}$$

As PIE fails to recover an item only when it cannot be recovered from any of its k mapped cells, the FNR is given by Equation (9). □

F. Proof of Theorem IV.2

Proof. There are two scenarios that give rise to false positives: (1) the recovered ID does not actually belong to any of the items seen at the observation point during the T measurement periods; and (2) the recovered ID is exactly the same as some other persistent item. Next, we discuss these two scenarios.

Scenario 1: The item ID must be recovered from encoded bits that come from at least two different items from two different cells. During the verification, PIE will first examine whether the hash-print of the recovered ID is consistent with the one stored in the cells that corresponds to at least two different items at the considered position. The probability of error here equals $1/2^p$. During the verification, PIE will also check the other $k-1$ cells for recorded Raptor codes and hash-prints. This verification step will fail if the other $k-1$ cells are all collided, where the probability of a cell not being collided was calculated in Equation (3) and is represented by P_{nc}. Thus, the probability of failure of this second verification step is $(1/2^{r+p})^{2(k-1)P_{nc}}$. The false positive rate for this scenario is given by the following equation.

$$P_{FP1} = \left(\frac{1}{2^p}\right)\left(\frac{1}{2^{r+p}}\right)^{2(k-1)P_{nc}} = 2^{-\{p+2(r+p)(k-1)P_{nc}\}}$$

Scenario 2: Suppose the recovered item ID corresponds to an item with occurrence t ($1 \leq t < T_{th}$), then there must be extra cells from other items that make the group mingled. Consider the worst case when there is only one other item making the group mingled. By similar analysis as for the first case, we can derive that the false positive rate for the second scenario is given by the following equation.

$$P_{FP2} = 2^{-\{r+(r+p)(k-1)P_{nc}\}} \tag{22}$$

Combining the analysis for these two scenarios, the FPR for the worst case is given by Equation (10). □

G. Proof of Lemma V.1

Proof. When m is fixed, the objective function in the Formulation 3 achieves its minimum value if p is maximized. Substituting the value of P_{nc} from Equation (3) and $r = M/m - p - 1$ from the lemma statement into the inequality in the lemma statement and rearranging, we get

$$p \leq \frac{M}{m} - 1 - \frac{\kappa l}{T_{th}} \times \left(1 - \frac{1}{m}\right)^{-\frac{kN}{T}} \tag{23}$$

This equation shows that the value of p is maximized when the inequality takes the *equal* sign. As the objective function in the third formulation is minimized when p is maximized, the FNR is minimized when the inequality takes the *equal* sign. □

H. Proof of Lemma V.2

Proof. Substituting m for $m = M/(r+p+1)$ from the lemma statement and P_{nc} from Equation (3) into the objective function in Formulation 3, the objective function becomes

$$\left(\frac{N}{2^p} \times \frac{(r+p+1)}{M}\right)^k \left(1 - \frac{r+p+1}{M}\right)^{k^2 N/T} \tag{24}$$

As both these terms are monotonically decreasing functions of p, the false negative rate is minimized when p is maximized.

Furthermore, after doing the same substitutions into the inequality in the lemma statement and rearranging, we get

$$p \leq M\left(1 - \left(\frac{\kappa l}{r \times T_{th}}\right)^{\frac{T}{kN}}\right) - 1 - r \tag{25}$$

Thus, the value of p is maximized when the inequality takes the equal sign, which in turn means that false negative rate is minimized when the inequality takes the equal sign. □

I. Proof of Lemma V.3

Proof. By taking first order derivative of the objective function in Formulation 3 with respect to m and equating it to zero, we observe that the objective function is maximized at $m = m_1^* = 1 + kN/T$. Furthermore, the value of the objective function increases monotonically in the range $[1, 1+kN/T)$ and then decreases monotonically in the range $[1+kN/T, +\infty]$. Substituting r by $r = M/m - p - 1$ from the lemma statement and P_{nc} from Equation (3) into the inequality in the lemma statement, we get

$$\left(\frac{M}{m} - p - 1\right)\left(1 - \frac{1}{m}\right)^{kN/T} \geq \frac{\kappa l}{T_{th}} \tag{26}$$

The left hand side of the inequality maximizes at $m = m_2^* = (1+kN/T)/(1+kN(p+1)/T/M)$. Furthermore, the inequality holds in a continuous range $[m_l, m_r]$ and both m_l and m_r make the above inequality take the equal sign. The objective objective function should be minimized at either m_l or m_r due to its monotonicity in the range $[m_l, m_r]$, which means that the inequality shown in the lemma statement should take the equal sign. □

J. Proof of Theorem V.1

Proof. We prove the sufficient condition for optimization by adopting the method of proof by contradiction. Suppose the false negative rate is minimized when the inequality shown in the theorem statement constraint does not take the equal sign. As per Lemmas V.1, V.2 and V.3, we can fix the value of any one parameter amongst m, r and p, and optimize the remaining two by letting the inequality in the theorem statement take the equal sign, to further minimize the objective function. This contradicts with the earlier supposition that false

negative rate is minimized when the inequality shown in the theorem statement does not take the equal sign. The expression for r in Equation (12) directly follows from the inequality in the theorem statement. Similarly, the expression for p in Equation (13) directly follows from the inequality in the theorem statement along with the expression $m(r+p+1) = M$ in the theorem statement. By plugging Equations (12) and (13) into the objective function in Formulation 3, we get

$$\left(2^{-\left(\frac{M}{m}-\frac{\kappa l}{T_{th}}(1-\frac{1}{m})^{-kN/T}-1\right)}\frac{N}{m}(1-\frac{1}{m})^{kN/T}\right)^{k} \quad (27)$$

Taking the first-order derivative of the function above and equating it to 0, we get the sufficient condition for optimization given in Equation (11) in the theorem statement. □

K. Proof of Theorem VI.1

Proof. As described earlier, X and Y are approximately independent and consequently, $Z = X + Y$. We can write the probability distribution of Z by enumerating all possible combination of X and Y for a fixed value of Z, *i.e.*,

$$P\{Z=z\} = \sum_{z_f=0}^{z}\Big\{P\{X=z_f\}\times P\{Y=z-z_f\}\Big\} \quad (28)$$

As both X and Y follow binomial distributions, substituting the expressions for their probability distributions in the equation above results in Equation (14). Furthermore, as the expected value and variance of the sum of two independent variables is simply the sum of their expected values and variances, respectively, the expressions for $E(Z)$ and $Var(Z)$ simply follow from this rule. □

L. Proof of Theorem VI.2

Proof. The probability of getting the observed value $\tilde{Z}[j]$ for some cell line is given by Equation (14). Consequently, the probability for all observed values in all cell lines to occur is

$$\begin{aligned} L &= \prod_{j=1}^{k} P\Big\{Z=\tilde{Z}[j]\Big\} \quad (29)\\ &= \prod_{j=1}^{k}\left[\sum_{z_f=0}^{\tilde{Z}[j]}\left\{\binom{\mathcal{N}_i}{z_f}(\mathcal{P}_{cs})^{z_f}(1-\mathcal{P}_{cs})^{\mathcal{N}_i-z_f}\times \right.\right.\\ &\quad \left.\left.\binom{\frac{kN}{T}}{\tilde{Z}[j]-z_f}(\mathcal{P}_{ms})^{\tilde{Z}[j]-z_f}(1-\mathcal{P}_{ms})^{\frac{kN}{T}-\tilde{Z}[j]+z_f}\right\}\right] \end{aligned}$$

For ease of analysis, we first apply logarithm to both sides of the above equation

$$\ln(L) = \sum_{j=1}^{k}\ln\left[\sum_{z_f=0}^{\tilde{Z}[j]}\left\{\binom{\mathcal{N}_i}{z_f}(\mathcal{P}_{cs})^{z_f}(1-\mathcal{P}_{cs})^{\mathcal{N}_i-z_f}\times\mathbf{F}(j)\right\}\right]$$

where

$$\mathbf{F}(j) = \binom{\frac{kN}{T}}{\tilde{Z}[j]-z_f}(\mathcal{P}_{ms})^{\tilde{Z}[j]-z_f}(1-\mathcal{P}_{ms})^{\frac{kN}{T}-\tilde{Z}[j]+z_f}$$

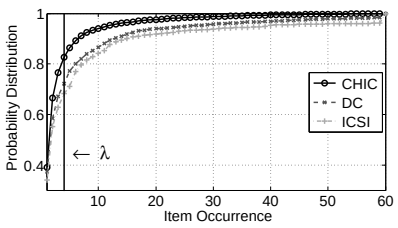


Fig. 5. Item occurrence prob. dist. for three traces

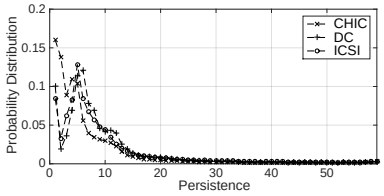


Fig. 6. Item persistence prob. dist. for three traces

As $\frac{d}{dv}\binom{v}{w} = \binom{v}{w}\big(\psi^{(0)}\{v+1\} - \psi^{(0)}\{v+1-w\}\big)$, where $\psi^{(0)}\{.\}$ is the 0^{th} order polygamma function [40], we represent

$$\begin{aligned} \mathbf{P}(j) &= \frac{d}{di}\left[\binom{\mathcal{N}_i}{z_f}(\mathcal{P}_{cs})^{z_f}(1-\mathcal{P}_{cs})^{\mathcal{N}_i-z_f}\right]\\ &= \binom{\mathcal{N}_i}{z_f}(\mathcal{P}_{cs})^{z_f}(1-\mathcal{P}_{cs})^{\mathcal{N}_i-z_f}\\ &\quad \Big(\psi^{(0)}\{\mathcal{N}_i+1\} - \psi^{(0)}\{\mathcal{N}_i+1-z_f\} + \ln(1-\mathcal{P}_{cs})\Big) \end{aligned}$$

Finally, by taking the first-order derivative of $\ln(L)$ and equating it to zero, we get Equation (17). According to the principle of the maximum likelihood estimation (MLE) method, the numerical solution of Equation (17) for $\mathcal{N}_i$ is the estimate $\tilde{\mathcal{N}}_i$. □

M. Tables and Figures for Evaluation

TABLE II
SUMMARY OF ITEM TRACES

Trace	Duration	# pkts	# flows
CHIC	6 mins	25.3M	101374
ICSI	1h	1.49M	8797
DC	1h	8.09M	10289

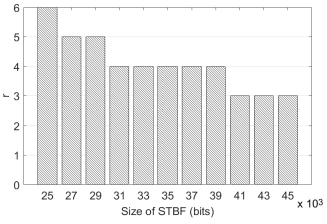


Fig. 13. Coding efficiency optimization

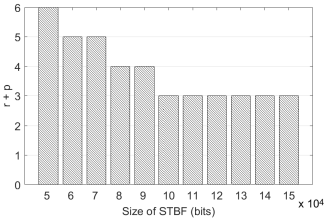
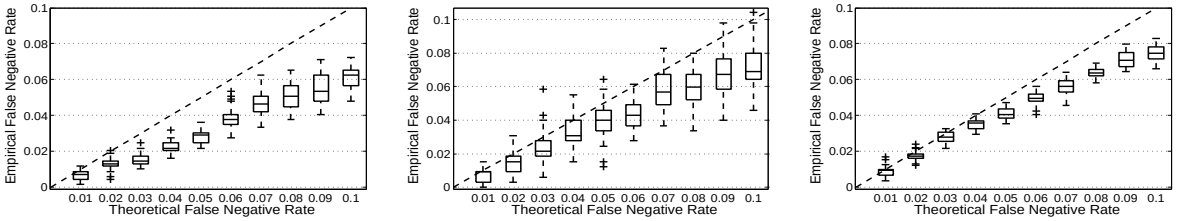
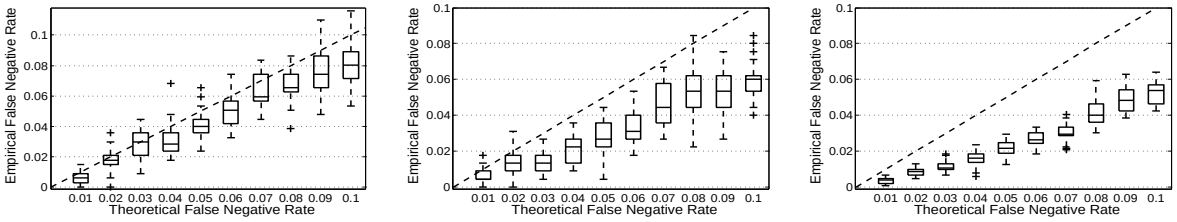


Fig. 14. Memory access efficiency optimization



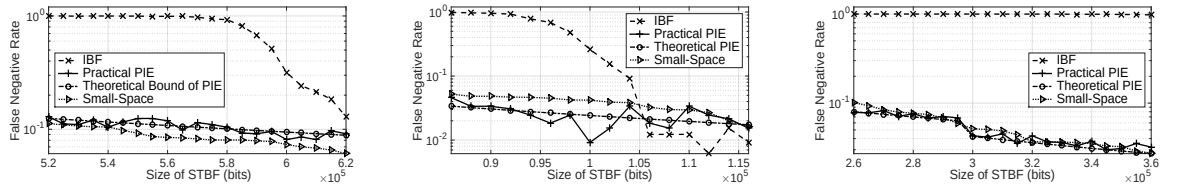
(a) CHIC (b) DC (c) ICSI

Fig. 7. Empirical false negative rate vs. theoretical false negative rate when $T_{th} = 40$



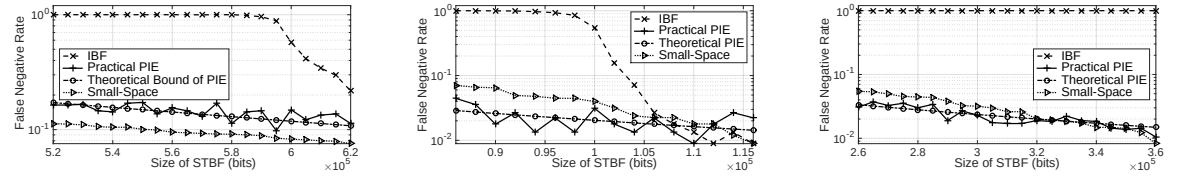
(a) CHIC (b) DC (c) ICSI

Fig. 8. Empirical false negative rate vs. theoretical false negative rate when $T_{th} = 50$



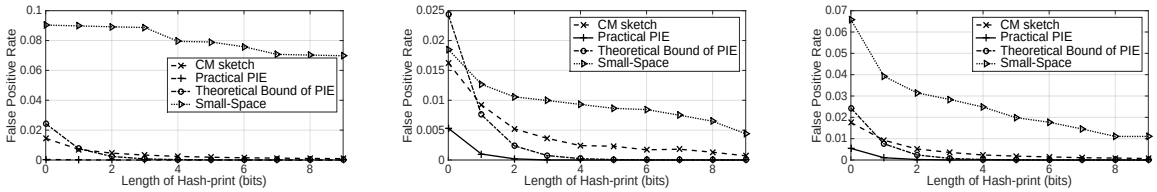
(a) CHIC (b) DC (c) ICSI

Fig. 9. False negative rate when $T_{th} = 40$



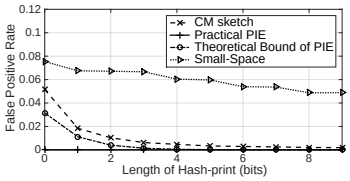
(a) CHIC (b) DC (c) ICSI

Fig. 10. False negative rate when $T_{th} = 50$

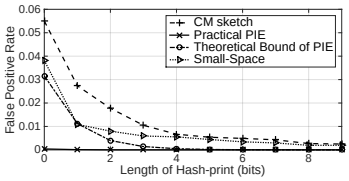


(a) CHIC (b) DC (c) ICSI

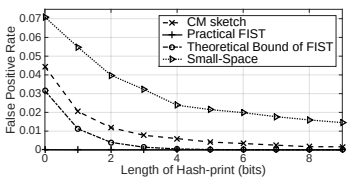
Fig. 11. False positive rate when $T_{th} = 40$



(a) CHIC



(b) DC



(c) ICSI

Fig. 12. False positive rate when $T_{th} = 45$

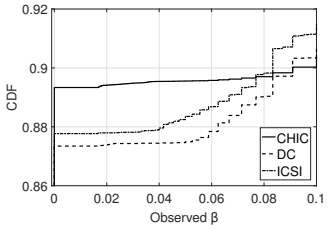


Fig. 15. CDF of observed β for $\alpha = 0.9$ and $\beta = 0.1$

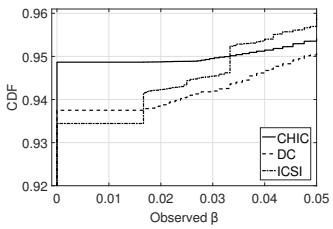


Fig. 16. CDF of observed β for $\alpha = 0.95$ and $\beta = 0.05$

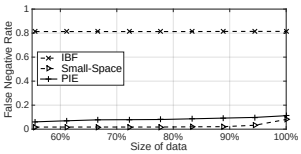


Fig. 17. FNR vs. data size

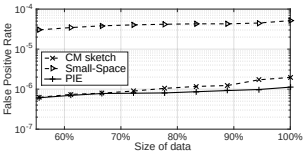


Fig. 18. FPR vs. data size

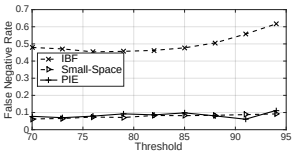


Fig. 19. FNR vs. T_{th}

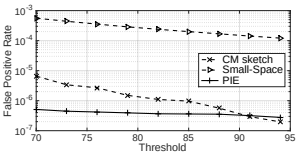


Fig. 20. FPR vs. T_{th}