Global Iceberg Detection over Distributed Data Streams

Haiquan (Chuck) Zhao*, Ashwin Lall*, Mitsunori Ogihara†, Jun (Jim) Xu*

* College of Computing, Georgia Institute of Technology
† Department of Computer Science, University of Miami

***Abstract*— In today's Internet applications or sensor networks we often encounter large amounts of data spread over many physically distributed nodes. The sheer volume of the data and bandwidth constraints make it impractical to send all the data to one central node for query processing. Finding distributed icebergs—elements that may have low frequency at individual nodes but high aggregate frequency—is a problem that arises commonly in practice. In this paper we present a novel algorithm with two notable properties. First, its accuracy guarantee and communication cost are independent of the way in which element counts (for both icebergs and non-icebergs) are split amongst the nodes. Second, it works even when each distributed data set is a stream (i.e., one pass data access only).**

Our algorithm builds upon sketches constructed for the estimation of the second frequency moment (F_2) of data streams. The intuition of our idea is that when there are global icebergs in the union of these data streams the F_2 of the union becomes very large. This quantity can be estimated due to the summable nature of F_2 sketches. Our key innovation here is to establish tight theoretical guarantees of our algorithm, under certain reasonable assumptions, using an interesting combination of convex ordering theory and large deviation techniques.

I. Introduction

Today's Internet applications often generate and collect a massive amount of data at many distributed locations. For example, an ISP (Internet Service Provider) security monitoring application may require that packet traces be collected at hundreds (or even thousands) of ingress and egress routers, and the amount of data collected at each router can be in the order of several terabytes. From time to time, various types of queries need to be performed over the union of these data sets. For example, in this ISP security monitoring application, we may need to query the union of packet trace data sets at all ingress and egress points to look for globally frequent signatures that may correspond to certain Internet worms. Given the gigantic and evolving nature of these physically distributed data sets, it is usually infeasible to ship all the data to a single location for centralized query processing due to the prohibitively high communication cost. Another scenario is that, in a sensor network, constraints on power consumption limit the amount of data that each sensor can transmit to a central server. Therefore, how to execute various types of (approximate) queries over the union of distributed data sets without physically merging them together has received considerable research attention recently.

One such distributed query problem that has been studied extensively is to detect global heavy-hitters or *icebergs*, which are data elements whose aggregate frequency across all these data sets exceed a pre-specified threshold. The hardness of this problem arises from the fact that a global iceberg may be finely distributed across all the measurement points so that it does not appear large at any one location. For example, in security scenarios an adversary may conceal the presence of the iceberg by spreading it thinly across many different nodes. This precludes the possibility of using a naive algorithm that simply reports locally frequent elements. On the other hand, it would be prohibitively expensive for every node to send records for every small fragment to the central server.

We propose a solution with the salient property that it is unaffected by the manner in which the data is distributed across the local nodes. To attain this property we use *summable sketches* that can be computed locally and later summed at a central location to answer queries on the aggregated data. Due to the nature of these sketches, it does not matter how the data was distributed among the nodes, or even in what order the data is aggregated, making the performance of our solution dependent solely on the aggregated data (and independent of how it is split among nodes). Also, the performance guarantee of these sketches is independent of the number of elements inserted into them, making them ideal for this problem.

Now, one longstanding issue with the iceberg detection problem is that it is notoriously difficult to handle elements that are close to the iceberg threshold. If there are many non-icebergs near the iceberg size, then it is virtually impossible for any approximate algorithm to distinguish the iceberg from the non-icebergs. A reasonable requirement for a data set to avoid this issue is that there is a gap between the size of icebergs and non-icebergs with only a few elements that fall within this gap. We call this the *sparsely populated gap* assumption. The analysis of our algorithm makes use of this assumption.

While requiring such a gap between icebergs and non-icebergs sounds restrictive, this assumption is actually quite practical in many real-world applications. This is because many data sources follow a power-law distribution in which the most frequent elements appear many times more often than the average frequency. For example, network data is commonly observed to follow such a heavy-tailed distribution, where extremely large flows are few and far between. It is critical to detect distributed icebergs in such data, e.g., when monitoring for a distributed denial of service attack, no single link may contain sufficient evidence of the attack to raise a flag.

We begin by studying the problem in which there are no elements in the gap. The analysis of our solution takes advantage of this gap and uses an asymptotically near-optimal amount of communication to solve this problem. This near-optimality is proven rigorously by comparing the communication complexity of our solution and the asymptotic communication complexity lower bound we establish for this problem. We then show how to reduce the size of the gap by using some additional information about the data. Finally, we discuss the effect of the elements in the sparsely populated gap.

We envision applications of our solution in which a central server is monitoring a large number of distributed nodes for large outliers. Since it is infeasible for all the data to be shipped to the central server, each local node sends a compact summary of its data to the central server in a single round. If the central server detects that there is an iceberg, it may initiate additional rounds of communication to confirm this fact.

Even though we describe our solution using this simple one-tier topology (a server communicating with many client nodes), we will later show how our solution can be very easily generalized to arbitrary tree topologies with identical communication costs and analytical guarantees. For example, this solution naturally fits the framework of Google MapReduce [1] and the Apache Hadoop architecture [2].

Very often, the data is not found aggregated on the nodes but is presented as a stream of updates (e.g., network packet data). In such cases, it is important to keep the processing requirements of the local algorithm low. Our solution works not only when the data is already locally aggregated (bag case) but also when it appears as a stream.

A. The "sketch" idea of our solution

We approach this problem by making use of *summable sketches* to succinctly encode the data at each local node. A summable sketch is a sketch (i.e., a lossy, succinct representation of a data set or stream) that has the following additional property: the sketch for the union of two data sets $A \cup B$ can be easily computed from the sketches of A and B. In our solution approach, each node computes the sketch of its local data set and ships it to the central server. The server will then in turn "sum up" these sketches to obtain the sketch for the union of the data set, which will be able to detect the global icebergs with high confidence.

The summable sketches we find most useful for our problem are those that compute the *second frequency moment* (i.e., the sum of the squares of the frequencies of all elements) or F_2 of the data aggregated across all the local nodes. The F_2 value is intuitively a good indicator of iceberg existence/nonexistence because of its "squaring effect" that significantly magnifies the skewness of the data (if any). For example, an iceberg item that is 100 times larger than a non-iceberg item contributes $100^2 = 10,000$ times more to the total F_2 value! Conceivably, we could have also used even higher frequency moments (say $F_3, F_4, F_5, \ldots$) to further magnify such skewness. However, it can be shown that estimating the k^{th} frequency moment ($k > 2$) incurs a minimum communication cost of $\Omega(n^{1-2/k})$ [3], where n is the total number of elements. In sharp contrast, the second frequency moment can be approximated using a sketch with size *independent* of n [4].

While techniques for estimating F_2 have been well-studied, our contribution lies in that (1) we successfully adapt them to the detection of global icebergs in a split-independent manner and (2) we are able to obtain very sharp accuracy bounds using an interesting combination of convex ordering and large deviation techniques.

Because we use summable F_2 sketches (e.g., Alon, Matias, and Szegedy's tug-of-war sketch [4]), our proposed algorithm has several desirable features that distinguishes it from prior work. First, it has the *split independence* property, i.e., both its performance guarantee and communication overhead are *independent of the way the total frequency of each and every element is split across the nodes*. We will show this is an immediate consequence of F_2 sketches being summable.

Second, since F_2 sketches were designed for streaming updates, our methodology works even when the local nodes have their data streamed to them at very high rates. This makes our algorithm more generally applicable than some previous work (e.g., [5]) that assumes the data is already aggregated without information loss at each local node (so-called "bag case").

Third, due to the summable nature of the sketch, we can handle arbitrary connected topologies among the nodes and the central server (e.g., flat topology in [5] and hierarchical tree topology in [6]) with the same accuracy and communication overhead guarantees. In other words, our algorithm is "oblivious" to the interconnection topology.

Furthermore, we show that once an iceberg is detected, we can estimate its size approximately with absolutely no additional communication overhead. In the "bag case" we can ascertain the precise size with an additional round of communication. But we show how to do away with this if we only want an approximate answer, making ours a one-round communication protocol.

The rest of this paper is laid out as follows. In Section II we formally define our problem. We describe our algorithm and the summable sketches upon which it is based in Section III. We completely solve a simplified version of our problem, with a gap assumption, in Section IV and show how the iceberg size can also be estimated at no additional cost. In Section V we use some additional information about the data to reduce the required magnitude of the gap. We finally discuss how our algorithm can be applied to real data in Section VI and highlight some of its useful properties. Our algorithms are evaluated experimentally using Internet flow data in Section VII. We describe some related works in Section VIII before concluding in Section IX.

II. Problem Definition

Consider a system or network that consists of m distributed nodes (e.g., routers). The data set S_j at node j contains a stream of tuples $\langle element\_id, c \rangle$, where $element\_id$ is an element identity from a set $\mathcal{U} = \{u_1, u_2, u_3, \ldots, u_n\}$ and c

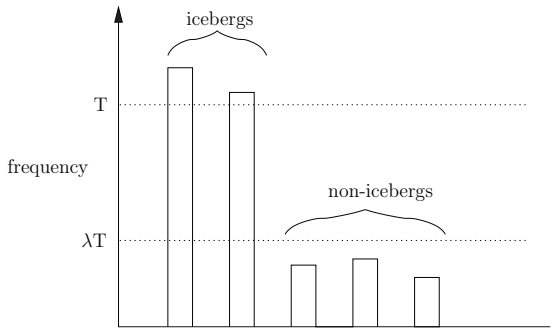

Fig. 1. The gap between icebergs and non-icebergs

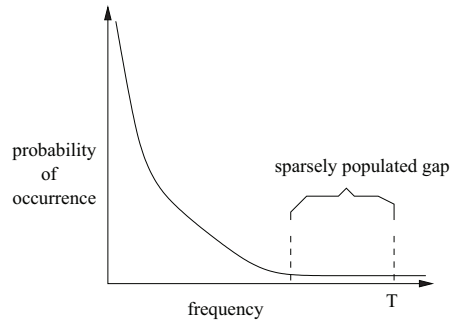

Fig. 2. Illustration of the sparse gap for real data sets

is an incremental count. We denote by $c_i = \sum_j \sum_{\langle u_i, c\rangle \in S_j} c$ the frequency of the element u_i when aggregated across all the nodes. We want to detect the presence of elements whose total frequency across all the nodes adds up to exceed a given threshold T. In other words, we would like to find out if there exists an element $u_i \in \mathcal{U}$ such that $c_i \geq T$. We desire our solution to be independent of how the elements are split among the nodes, i.e., our final solution should be dependent on $c_1, \ldots, c_n$, but not on how each c_i is split among the m nodes.

In most iceberg detection scenarios, it is critical to discover the iceberg every time. Hence, we will err on the side of caution by having almost no false negative error even if this means being more permissive to false positive error.

Now, the main issue that we face is that any element that is slightly under the threshold will be nearly indistinguishable from an iceberg. To get around this problem, we will make some simplifying assumptions on the size of non-icebergs and then later demonstrate how these assumptions can be weakened.

The first simplifying assumption that we make is that it is guaranteed that the iceberg is much larger than any non-iceberg. More formally, we say that an element whose aggregate frequency is at least T is an iceberg, and we assume that no element has aggregate frequency in the interval $(\lambda T, T)$, for some $\lambda \in (0, 1)$ (illustrated in Figure 1). This gap parameter λ is independent of n (number of elements) and m (number of nodes).

The gap assumption may be reasonable in certain security scenarios in which massive icebergs are hidden among the many nodes by an adversary. For example, a DDoS attacker may mount an attack that results in the victim receiving hundreds of times more traffic than any other host while spreading this traffic thinly across many different paths to avoid detection. Similarly, a network worm may attempt to avoid detection by spreading very slowly at any single point, even though it has massive aggregate volume.

However, not in all scenarios can we make such a gap assumption. Additionally, even if there is a gap, we may not know how large it is *a priori*. To deal with this issue, we will later weaken this assumption to allow some elements (though not many) to enter this gap. This is reasonable because it is commonly observed in real data that the occurrence of high-frequency items rapidly tails off. We call this a sparsely populated gap (see Figure 2). Our ultimate goal will be to solve the problem of detecting icebergs in real data that exhibits the sparsely populated gap property.

III. Algorithmic Overview

To solve this problem, we use a *summable sketch*. Summable sketches have the property that we can "sum" the sketches from the individual nodes together to get an aggregate sketch that is identical to the sketch of the aggregate frequencies. This property allows us to guarantee that no matter how the iceberg and non-icebergs are distributed among the nodes, the result of our algorithm will always be the same.

For our solution, we use the sketch for the second frequency moment of the data, defined below.

Definition 1: The second frequency moment (F_2) of a data set is the sum of the squares of the frequencies of each item in the set. That is, $F_2(X) = \sum_{x \in X} freq(x)^2$.

There is typically a gap separating icebergs from non-icebergs in real data. By focusing on the second moment, we magnify this gap to make the difference even easier to detect.

We use F_2 sketches for estimating the second frequency moment for the following reasons:

1) The second moment makes extremal values (i.e., icebergs) stand out distinctly. Intuitively, if we could compute the higher moments (e.g., the tenth frequency moment), then we could further exaggerate this effect. As discussed earlier, however, computing higher frequency moments is much more expensive.
2) We found that some of the existing F_2 sketches for estimating the second moment have the aforementioned summable property. We show in this paper how this property can be exploited for the purpose of iceberg detection.
3) Additionally, the error analysis for these sketches is *independent of the number of elements inserted into it*, which allows us to fix error parameters without a need to account for n, the total number of elements.
4) Finally, the F_2 sketches were designed to be extremely cheap to update. Our solution is viable even if the local nodes process elements as *streams* of updates.

Algorithm 1 LOCAL SKETCHING ALGORITHM

PRE-PROCESSING:

Initialize g F_2 sketches $S_1, \ldots, S_g$.
Initialize hash function $h : \mathcal{U} \to \{1, \ldots, g\}$.

ALGORITHM:

for each element/frequency pair $\langle id, count \rangle$ **do**
  Insert id with frequency $count$ into the sketch $S_{h(id)}$.
end for

Algorithm 2 CENTRAL AGGREGATION ALGORITHM

PRE-PROCESSING:

Receive sketches $S_1^i, \ldots, S_g^i$ from each local node i.

ALGORITHM:

Sum sketches from each node to create aggregate sketches $S_1^*, \ldots, S_g^*$
for $i := 1$ to g **do**
  Estimate $F_2(S_i^*)$.
  if $F_2(S_i^*) \geq T^2(1-\epsilon)$ **then**
    Output "There is an iceberg present (at least one of $h^{-1}(i)$ is an iceberg)."
  end if
end for

The F_2 sketches we consider enable computation of the second moment with arbitrary precision and confidence. For all $\epsilon, \delta < 1$, these sketches can guarantee an ϵ relative error approximation with probability at least $1 - \delta$ using at most $O(\log(1/\delta)/\epsilon^2)$ counters, which is the asymptotically optimal number [4]. Note that the number of counters necessary is independent of the number of elements that are inserted into the sketch, which is a key property that we need.

A. Our Algorithm

Our algorithm works by randomly partitioning all the elements, uniformly at random, into groups and estimating F_2 for each of these groups. Any group with an iceberg in it will stand out from the rest because of its large F_2. On the other hand, there will be few false positives because non-icebergs are usually much smaller. For example, if the iceberg is ten times larger than most other elements, then one hundred separate non-icebergs would have to fall into a group to make it appear to have an iceberg. We give a more formal description of the algorithm next.

We partition the elements in $\mathcal{U}$ into g groups using a hash function, $h : \mathcal{U} \to \{1, 2, 3, \ldots, g\}$, which is shared by all the nodes. Each node creates a separate F_2 sketch for the elements of each of these groups and updates them over the stream. At the conclusion of the stream (or at regular intervals for infinite streams), each node sends all of its sketches to the central server. See Algorithm 1.

The central server sums the sketches for each of the g groups and obtains an approximation of the second moment for each of these groups. If any group has estimated F_2 over $(1-\epsilon)T^2$, the algorithm signals that there is an iceberg present. (See Algorithm 2.) For each such group, the central server can poll the nodes for the exact counts for that group. Alternatively, this procedure can be repeated recursively on the suspect group until the iceberg is identified.

Our algorithm has a low false negative rate. The estimate of F_2 for any group with an iceberg in it will be at least $(1-\epsilon)T^2$ assuming that the F_2 sketch for that group did not err with greater than ϵ relative error—this happens with probability at least $(1 - \delta)$. As a result, we can keep the false negative rate as low as we desire simply by ensuring that the sketches have a suitable small failure rate δ.

In the following section we briefly describe the F_2 sketch of Alon, Matias, and Szegedy [4] and describe how it has all the desirable properties that we require.

B. The Tug-of-War Sketch

As part of our solution, we make use of the Tug-of-War Sketch Algorithm, introduced by Alon, Matias, and Szegedy [4]. The tug-of-war sketch is a means of summarizing frequency data in a stream so that the second moment of the frequencies can be computed efficiently from it. This sketch allows for arbitrary updates (i.e., we may increment the frequency of an element by an arbitrary integer) and is very fast to update.

The tug-of-war sketch enables computation of the second moment with arbitrary precision and confidence, i.e., for all $\epsilon, \delta < 1$, the sketch can guarantee an ϵ relative error approximation with probability at least $1 - \delta$ using at most $O(\log(1/\delta)/\epsilon^2)$ counters. Below, we will briefly describe how it works.

The tug-of-war sketch computes $z = 32\log(1/\delta)/\epsilon^2$ unbiased estimates for the second moment as follows. Each estimate is the linear projection of the frequencies multiplied by coefficients ± 1, which are computed from hash functions of the form $h : \mathcal{U} \to \{-1, 1\}$. It can be shown that, by choosing h to be 4-wise independent, the square of this sum is an unbiased estimate of the second moment. These estimators are then divided into groups and the median of the averages of the groups can be shown to be an extremely robust estimate of the second moment.

The tug-of-war sketch uses just $O(\log(1/\delta)/\epsilon^2)$ estimators—the asymptotically optimal number [4]—which bounds both the number of counters needed by it and the number of operations needed to update it. This makes it very efficient to update in a stream. Note that the number of counters necessary is independent of the number of elements that have been inserted into the sketch, which allows us to use the same sketch size for each distributed node and group.

Each estimator of the tug-of-war sketch is a linear projection of the form $\vec{c} \cdot \vec{v} = c_1 v_1 + \ldots + c_n v_n$, where $\vec{c}$ is the vector of frequencies and $\vec{v}$ is a vector in $\{-1, 1\}^n$. This permits arbitrary updates to the sketch (i.e., updates with both positive and negative integers) since updating the frequency of the ith element by u can be done by simply adding uv_i to the estimator. Additionally, if two sketches use the same hash

functions (i.e., the same vector $\vec{v}$), they can be directly summed to give the sketch that would have resulted from taking the union of the original inputs. *This extremely powerful summable property is what allows us to aggregate the result of the nodes in a split-independent fashion.*

We note that Indyk's stable distribution sketch [7] also has the same desirable properties as the tug-of-war sketch. Namely, it is summable, is efficient to update, and has the same asymptotic space bound. However, in practice the stable distribution sketch needs considerably more space than the tug-of-war sketch because of its requirement of independent, stably-distributed values [7].

IV. The Gap Assumption

One issue that our algorithm, and indeed any approximate algorithm for this problem, must overcome is that it is virtually impossible to distinguish an iceberg from any non-icebergs close to its size. To assist with this issue, we introduce the concept of the gap assumption.

The gap assumption is an assumption that we make about the measured data to assist in correctly detecting icebergs. According to this assumption, there will never be any non-icebergs in the range $(\lambda T, T)$, where T is the threshold for icebergs and $\lambda \in (0, 1)$ is a known gap parameter. In this section we will assume this assumption to be strictly true, and later we will discuss the effect of having a few non-icebergs in the gap.

We will show that our summable sketch-based methodology for iceberg detection results in an asymptotically near-optimal algorithm for the gap assumption problem. We do so by first demonstrating a lower bound for this problem and then demonstrate how our algorithm nearly matches this lower bound.

A. Lower Bound for the Gap Assumption Problem

Consider the following game played among s players: Suppose that each of the s players has a set of t elements from the universe $\{1, \ldots, n\}$, where $n = (2t-1)s$. Call these sets $A_1, \ldots, A_s$. We are guaranteed that either one of the following two situations are true:

- For all $i, j \in \{1, \ldots, s\}$ with $i \neq j$, $A_i \cap A_j = \emptyset$.
- For all $i, j \in \{1, \ldots, s\}$ with $i \neq j$, $A_i \cap A_j = \{x\}$, for some $x \in \{1, \ldots, n\}$.

That is, it is either the case that all the sets are pairwise disjoint, or it is the case that all the sets have precisely one element in common. The problem is then for these s players to determine which of the above two situations is true (their behavior when neither case holds can be arbitrary).

This problem was originally studied by Alon, Matias, and Szegedy [4] in the context of proving lower bounds for the estimation of the frequency moments, and they gave an $\Omega(n/s^4)$ bound. This bound was subsequently improved to $\Omega(n/s^2)$ by Bar-Yossef *et al.* [8] and finally to $\Omega(n/(s \log s))$ by Chakrabarti, Khot, and Sun [3]. We make use of this final result.

Theorem 1: Any algorithm for detecting an iceberg that is at least $1/\lambda$ times the size of the next largest element will require each node to communicate $\Omega(n\lambda^2/\log(1/\lambda))$ bits on average.

Proof: Consider the special case where we are guaranteed that it is either the case that all the s nodes have pairwise disjoint sets of identities or all of them have precisely one iceberg in common. By the result of Chakrabarti, Khot, and Sun [3], we know that such an algorithm must communicate at least $\Omega(n/(s \log s))$ bits of information. Hence, on average, each node communicates $\Omega(n/(s^2 \log s))$ bits (and, in particular, some node must communicate at least so much information).

For this problem, the λ guarantee we are provided is $1/s$ since the iceberg has size s (i.e., the number of nodes) and every other element has size at most 1. Hence, for this class of problem instances, $s = 1/\lambda$. Substituting this into the lower bound from above, we get that on average each node communicates at least $\Omega(n\lambda^2/\log(1/\lambda))$ bits. ∎

Note that this lower bound shows a result stronger than what we had set out to achieve: this bound applies even in the case where each element appears at each location with frequency either 0 or 1, and any protocol for point-to-point communication between the nodes is used.

B. Algorithm for the Gap Assumption Problem

We now show that our algorithm is able to achieve a communication cost of $O(n\lambda^2)$. Since we showed in the previous section that the lower bound for this problem is $\Omega(n\lambda^2/\log(1/\lambda))$, this solution is near-optimal. In fact, we believe $O(n\lambda^2)$ to be a tight bound and conjecture that the lower bound can be strengthened to remove the $\log(1/\lambda)$ factor.

Our solution for this problem is to simply use Algorithm 1 with $g = 6n\lambda^2$ as the number of groups. In the following sections we prove the communication cost bounds for this algorithm and give analysis showing its accuracy in correctly detecting the presence of icebergs.

1) Communication Cost: Since each sketch has cost $O(\log(1/\delta)/\epsilon^2)$, our algorithm requires each local node to communicate a total of $O(g \log(1/\delta)/\epsilon^2)$ counters to the central server. Taking constant ϵ and δ, we have that the communication cost of our algorithm is $O(g) = O(n\lambda^2)$.

In comparison, while the naive algorithm has each local node send a counter for each element (for a total of $(1 + \Omega(1))n$ counters), our algorithm requires $192n\lambda^2 \log(1/\delta)/\epsilon^2$ counters, which gives us large savings when λ is small (e.g., 1/1000).

The tug-of-war sketch can be modified to use $2/(\delta\epsilon^2)$ counters by just averaging the estimates, rather than taking the median of averages. This is less than $32 \log(1/\delta)/\epsilon^2$ when δ is not too small. For example, if we take $\epsilon = 1/2, \delta = 0.05$, then our algorithms requires $960n\lambda^2$ counters, which gives us considerable savings when λ is as large as $1/100$.

Note that since our algorithm does not need to send the identities of elements along with their counts, we are not burdened with this additional overhead. A naive method, on

the other hand, necessarily must transmit element identities to aggregate the counts of all the elements. Hence, our algorithm will especially shine when element identities are large (e.g., IP flow labels).

Numerical Example: Consider a situation each distributed node has $m = 1000000$ (one million) search queries that they need to communicate to the central server. Let us assume that each element has an identity of size 12 bytes and a counter of size 8 bytes. Further, let us assume that we are guaranteed a gap of $\lambda = 1/100$ in the data. Then, a naive solution would require about $20 \times n = 20$MB of communication to identify any icebergs in the data. In contrast, our algorithm would need only $8 \times 960n\lambda^2 = 768$KB of communication to solve the problem. This gives us over an order of magnitude savings in the communication cost.

2) Analysis: We showed in the previous section that the false negative rate of our algorithm is determined solely by the failure rate of the sketches. By keeping this rate δ small, we will almost never miss a true iceberg. Hence, all that is left for us to show is that it is unlikely for a group without any icebergs in it to be a false positive.

Theorem 2: For every group with no iceberg in it, the iceberg detection algorithm erroneously signals that it has an iceberg in it with false positive probability at most $\delta+\delta'$, where δ is the failure probability bound of the tug-of-war sketch, $\delta' = (\frac{e}{4})^{1/(6\lambda^2)}$, and λ is the gap parameter.

Proof: To simplify our analysis, we consider the worst case input for our algorithm: when all n elements have count λT. It is not hard to see that if the non-icebergs are smaller than λT this will only decrease the probability of a false positive.

Since the sketches may err with ϵ relative error, a non-iceberg may appear to contribute as much as $T^2\lambda^2(1+\epsilon)$ to the measured F_2. As the threshold of detection is set to $T^2(1-\epsilon)$, a false positive could only occur when at least $\frac{T^2(1-\epsilon)}{T^2\lambda^2(1+\epsilon)} \geq 1/(3\lambda^2)$ non-icebergs get put in the same group, where we assume that $\epsilon \leq 1/2$. Let us denote by X the random variable indicating how many non-icebergs get put in one particular group and bound the probability of the event that X exceeds $1/(3\lambda^2)$.

Let us denote by X_i the event that element u_i is in our group, for $i \in \{1, \ldots, n\}$, so that $X = \sum_{i=1}^{n} X_i$. Clearly, X_i's are i.i.d. Bernoulli random variables with probability $1/g$, since an element may go into any group with equal likelihood. This permits us to use the Chernoff bound:

Theorem 3 (Chernoff Bound): Let $X_i, 1 \leq i \leq n$ be i.i.d. Bernoulli random variables with probability p, $X = \sum_{i=1}^{n} X_i$. For $\beta > 1$,

$$\Pr[X \geq \beta pn] < \left(\frac{e^{\beta-1}}{\beta^\beta}\right)^{pn}.$$

Applying the above Chernoff bound, we get the following

$$\begin{aligned}\Pr[X > 1/(3\lambda^2)] &= \Pr[X > 2(n/g)] \\ &\leq \left(\frac{e^{2-1}}{2^2}\right)^{n/g} = \left(\frac{e}{4}\right)^{1/(6\lambda^2)}.\end{aligned}$$

Since the error in the estimate occurs with probability at most δ, the false positive probability in question is at most $\delta + \delta'$, as desired. ■

Since we expect our algorithm to work only when $\lambda \ll 1$, we expect the δ term to dominate this failure probability. Not only does the above algorithm detect the presence of one or more icebergs, it narrows down the iceberg to a subgroup of the universe. Each group that is above the threshold can be polled to identify the iceberg. Since there are only an expected $1/(6\lambda^2)$ elements in each group, this cost is far lesser than that of sending frequencies of all n elements.

Numerical example: When $\lambda = 0.1$, the false positive probability for a group is at most 0.16%. For $\lambda = 0.05$, this probability drops to less than one in hundred billion (10^{-11}). Clearly, this probability is much smaller than the failure probability of the sketch, δ, which we take to be around 1% in practice. At worst, we expect 1% of the groups (and hence about 1% of the elements) to signal a false positive, which takes very little additional communication to drill down.

C. Estimating Iceberg Size

Besides detecting the presence of an iceberg, it would be useful to get an estimate on its size. Size information is useful in diagnosing the extent of the anomaly and could help in determining what action should be performed next. In this section we show how our solution allows us to obtain an approximate estimate of the actual size of the iceberg in this setting *without any additional communication overhead*. If this estimate indicates a severe problem, a more accurate (but expensive) estimate of the size of an iceberg can be computed using an additional round of communication.

1) Biased Estimator: The first algorithm for estimating the size of the detected iceberg is simple. We take the estimate of F_2 for the group the iceberg was found in and use the square root of this value as an estimate of the iceberg size. There are two sources of error for this estimate: the approximation of the F_2 estimation as well as the collision of non-icebergs in the same group. (We assume that the number of icebergs is small enough that no two icebergs get mapped to the same group with high probability.)

Suppose that we detect an iceberg of size $S \geq T$ in a group that has estimated F_2 above the $T^2(1-\epsilon)$ threshold. We first estimate by how much we may under-estimate its true frequency: this is bounded by the error of the F_2 estimation. Hence, with probability at least $1-\delta$, this algorithm returns an estimate $\hat{S}$ such that

$$\hat{S} \geq S\sqrt{1-\epsilon}.$$

Assuming that $\epsilon \leq 1/2$ (as earlier) we get the guarantee that $\hat{S} \geq S/\sqrt{2}$.

The analysis for the bound on over-estimating the size of the iceberg is slightly more involved since we now have to account for the collisions of non-icebergs in the same group. We start by bounding the probability that the collisions exceed the threshold $T^2/3$. As in the earlier detection analysis, this would require more than $\frac{T^2/3}{T^2\lambda^2} \geq \frac{1}{3\lambda^2}$ non-icebergs to be

in the same group as the iceberg. As earlier, we bound this probability:

$$\Pr[X > 1/(3\lambda^2)] < \left(\frac{e}{4}\right)^{1/(6\lambda^2)} = \delta'.$$

Since $S \geq T$, the total overestimation error for the F_2 estimation is at most $(T^2/3 + S^2)(1+\epsilon) - S^2 \leq S^2$ (again, where we assume $\epsilon \leq 1/2$).

Hence, we finally have the following theorem:

Theorem 4: With probability at least $1 - \delta - \delta'$ we can estimate the true size of an iceberg of size S by $\hat{S}$ with the guarantee that $S/\sqrt{2} \leq \hat{S} \leq S\sqrt{2}$.

Obtaining a more accurate estimate of the iceberg size comes at a higher cost. We may obtain much more accurate estimates for the size of the iceberg using smaller ϵ. However, recall that the dependence of the communication cost on ϵ is $1/\epsilon^2$, which grows very rapidly. For example, halving the relative error results in quadrupling the communication cost. Still, there is a natural tradeoff here.

2) Unbiased Estimator: The above estimator has the disadvantage of being biased by the collision of non-icebergs with the iceberg. A more accurate way to approximate the size of the iceberg is to estimate the mass of non-icebergs that collide with it and remove this from our estimate. We can use the average of the F_2 estimates of the other groups for this purpose. Here we show that this estimator is unbiased when there is only one iceberg in the whole dataset and the F_2 sketch used is unbiased (e.g., the tug-of-war sketch).

We will use Y_0 for the F_2 of the non-icebergs in the group with the iceberg, and $Y_1, ..., Y_{g-1}$ for the F_2 of all the other groups. We know that $E[Y_0] = E[Y_1] = ... = E[Y_{g-1}]$. Let Π be the (non-real-valued) random variable denoting how the n elements are placed into g groups. Assume the iceberg size is S. So the estimator for iceberg size is $\widehat{S+Y_0} - \frac{1}{g-1}\sum_{j=1}^{g-1}\widehat{Y_i}$, where the hat reflects the F_2 sketch estimator. So we have

$$\begin{aligned}
& \mathrm{E}[\widehat{S+Y_0} - \frac{1}{g-1}\sum_{j=1}^{g-1}\widehat{Y_i}] \\
= \; & \mathrm{E}_\Pi\left[\mathrm{E}[\widehat{S+Y_0} - \frac{1}{g-1}\sum_{j=1}^{g-1}\widehat{Y_i} \mid \Pi]\right] \\
= \; & \mathrm{E}_\Pi\left[S + Y_0 - \frac{1}{g-1}\sum_{j=1}^{g-1} Y_i\right] \\
= \; & S + \mathrm{E}_\Pi[Y_0] - \frac{1}{g-1}\sum_{j=1}^{g-1}\mathrm{E}_\Pi[Y_i] = S.
\end{aligned}$$

The first equality is due to the law of total expectation. The second equality is because we are using an unbiased F_2 sketch. The third equality is due to the linearity of expectations. Hence, the estimate using this method is an unbiased estimate of the actual iceberg size.

V. Using F_1 Information

In the previous section, we show that in the case of a strong gap assumption (e.g., λ values as small as 1/100), the tug-of-war sketch allows us to identify icebergs with high probability in one round. We also prove a lower bound communication complexity under these conditions, which show that our detection algorithm is indeed asymptotically optimal, although the result is not practically satisfying.

Fortunately, this is not the end of story. We are able to significantly improve our result, in terms of the required λ, by asking for one additional piece of information that can be obtained with very little cost. This additional piece of information is the sum of the counts (F_1) of all the elements across all the nodes. It can be obtained by asking every node to send in the sum of the counts of all its items. Adding them up at the central server results in the F_1 of the union of the dataset. The additional communication cost is simply a few more bytes per node. However, strictly speaking, this adds one more round to the protocol as follows. In the first round, the total local counts are sent to the central server and summed to get F_1. Then the optimal number of groups is computed based on this count, and this is broadcast to all the nodes. Finally, the nodes send grouped tug-of-war sketches to the server.

In reality, however, this additional round can be avoided in continuous monitoring applications when the change from one time window to the next is not gigantic. We can simply use the average F_1 of the past windows as the estimate of the F_1 of the next window, for the purpose of determining the optimal number of groups. Then the total count for the next window can come in together with the tug-of-war sketches in one round. This scheme will work because the accuracy of our detection is not sensitive to the number of groups as computed from F_1.

Readers may now wonder why this total count alone makes such a huge difference. This is because earlier we only knew that all the items had frequencies between 0 and $B = \lambda T$. Our false positive bound has to assume the worst case from this very large family. However, once we know F_1, there is a constraint on the item counts, resulting in a much smaller family of counts. The worst case of the smaller family is much better than the worst case of the larger family. However, the worst case of the larger family is mathematically trivial, i.e., every element has count B, which is what we considered in the previous section. Obtaining or even approximating the worst case for the smaller family, however, turns out to be extremely challenging mathematically.

One contribution of this work is to bound the worst case of this smaller family using a combination of convex ordering theory and large deviation theory. For bounding the false positive rate, we are interested in the second moment F_2 of any group without an iceberg. Each element has an independent and identical probability to fall into this group. The element counts vector $\{c_i\}$ is only subject to two constraints: $c_i \leq B, \forall i$, and $\sum_i c_i = L$, where L is the aggregate F_1. Since our scheme has to work with all possible element counts

vectors, our bound clearly has to be the worst case (i.e., the maximum) bound over all of them. However, the space of all such vectors is so large that enumeration over all of them is computationally prohibitive and low complexity optimization procedures in finding the worst case do not seem to exist.

Fortunately, we discover that the element counts vectors are dominated by a particular (i.e., worst-case) element counts vector, in the *increasing convex order* (not in the *stochastic order*). Since the moment generating function (MGF) of any random variable is an increasing convex function, we are able to dominate the MGFs of the F_2 of the group under all element count vectors by that under the worst-case vector. The final tail bound is obtained by simply applying the Chernoff bound to this worst-case MGF.

Let us denote by X_i the event that element u_i is in the chosen group. X_i's are i.i.d. Bernoulli random variables with probability $1/g$. Let X_c be the F_2 of the elements in this group, i.e. $X_c = \sum_{i=1}^{n} c_i^2 X_i$. We denote it X_c to emphasize that its distribution depends on the vector c. We want to bound the probability that $X_c > \frac{T^2(1-\epsilon)}{1+\epsilon} \equiv A$.

In the following, we first describe the standard Chernoff method of obtaining sharp tail bounds from the MGF of a random variable (in this case X_c):

$$\begin{aligned} \Pr[X_c > A] &= \Pr[e^{\theta X_c} > e^{\theta A}] \\ &\leq \frac{\mathrm{E}[e^{\theta X_c}]}{e^{\theta A}}, \end{aligned}$$

where $\theta > 0$ is any constant, and the last step is due to the Markov inequality.

Since this is true for all θ, we have

$$\Pr[X_c > A] \leq \min_{\theta > 0} \frac{\mathrm{E}[e^{\theta X_c}]}{e^{\theta A}}. \tag{1}$$

Then, we aim to bound the moment generating function $\mathrm{E}[e^{\theta X_c}]$ by finding the worst-case element count vector.

Since convex ordering techniques and related concepts are needed to establish the bound, we first present a few definitions here:

Definition 2 (Majorization [9, 1.A.1]): For any n-dimensional vectors a and b, let $a_{[1]} \geq \ldots \geq a_{[n]}$ denote the components of a in decreasing order, and $b_{[1]} \geq \ldots \geq b_{[n]}$ denote the components of b in decreasing order. We say a is *majorized* by b, denoted $a \leq_M b$, if

$$\begin{cases} \sum_{i=1}^{k} a_{[i]} \leq \sum_{i=1}^{k} b_{[i]}, & \text{for } k = 1, \ldots, n-1 \\ \sum_{i=1}^{n} a_{[i]} = \sum_{i=1}^{n} b_{[i]} \end{cases} \tag{2}$$

Definition 3 (Schur-convex [9, 3.A.1]): A function $f : \mathcal{R}^n \to \mathcal{R}$ is called *Schur-convex*, if $x \leq_M y$ implies $f(x) \leq f(y)$.

Definition 4 (Exchangeable random variables): A sequence of random variables $X_1, \ldots, X_n$ is called *exchangeable*, if for any permutation $\sigma : [1, \ldots, n] \to [1, \ldots, n]$, the joint probability distribution of the permuted sequence $X_{\sigma(1)}, \ldots, X_{\sigma(n)}$ is the same as the joint probability distribution of the original sequence.

For example, a sequence of independent and identically distributed random variables are exchangeable.

Definition 5 (Convex function): A real function f is called *convex*, if $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ for all x and y and all $0 < \alpha < 1$.

Definition 6 (Convex order [10, 1.5.1]): Let X and Y be random variables with finite means. Then we say that X is less than Y in (increasing) convex order, written $X \leq_{cx} Y$ ($X \leq_{icx} Y$), if $\mathrm{E}[f(X)] \leq \mathrm{E}[f(Y)]$ holds for all real (increasing) convex functions f such that the expectations exist.

Since the MGF ($\mathrm{E}[e^{X\theta}]$) is expectation of an increasing convex function ($e^{x\theta}$) of X, establishing increasing convex order will help to bound the MGF.

The following theorem from Marshall [9] about convex functions and exchangeable random variables has many useful corollaries.

Theorem 5 ([9, 11.B.1]): Let $X_1, \ldots, X_n$ be exchangeable random variables. Let $\Phi : \mathcal{R}^{2n} \to \mathcal{R}$ be a function of two vector arguments. Suppose that Φ satisfies

(i) $\Phi(x; a)$ is convex in a for each fixed x,
(ii) $\Phi(x\Pi; a\Pi) = \Phi(x; a)$ for all permutation matrices Π,
(iii) $\Phi(x; a)$ is Borel measurable in x for each fixed a.

Then $\Psi(a) = \mathrm{E}[\Phi(X; a)]$ is symmetric and convex.

Now we can prove the following theorem:

Theorem 6: Let g be a convex function. Let $X_1, \ldots, X_n$ be exchangeable random variables that take only non-negative values. Then $a \leq_M b$ implies $\sum_{i=1}^{n} g(a_i) X_i \leq_{icx} \sum_{i=1}^{n} g(b_i) X_i$.

Proof: Let f be any increasing convex function. Let $\Phi(x; a) = f(\sum_{i=1}^{n} g(a_i)|x_i|)$. We will verify that $\Phi(x; a)$ satisfies the conditions in Theorem 5. When x is fixed, $g(a_i)|x_i|$ is convex because $|x_i| \geq 0$ and g is convex. So $\sum_{i=1}^{n} g(a_i)|x_i|$ is a sum of convex functions, thus convex [11, Theorem 5.2]. So $\Phi(x; a)$ is a composition of an increasing convex function with a convex function, thus convex [11, Theorem 5.1], therefore (i) holds. (ii) obviously holds. When a is fixed, $\Phi(x; a)$ is continuous because f is necessarily continuous, so (iii) holds.

Therefore Theorem 5 tells us that $\mathrm{E}[\Phi(X; a)]$ is symmetric and convex, thus Schur-convex [9, 3.C.2]. $\mathrm{E}[\Phi(X; a)] = \mathrm{E}[f(\sum_{i=1}^{n} g(a_i)|X_i|)] = \mathrm{E}[f(\sum_{i=1}^{n} g(a_i) X_i)]$, due to the assumption $X_i \geq 0$. By definition of Schur-convexity, $a \leq_M b$ implies $\mathrm{E}[f(\sum_{i=1}^{n} g(a_i) X_i)] \leq \mathrm{E}[f(\sum_{i=1}^{n} g(b_i) X_i)]$. Since this is true for any increasing convex function f, by definition of increasing convex order we have $\sum_{i=1}^{n} g(a_i) X_i \leq_{icx} \sum_{i=1}^{n} g(b_i) X_i$. ■

Remark: Note that stochastic order does not hold here in general. Suppose $a_1 = a_2 = 1, b_1 = 0, b_2 = 2, g(x) = x^2$, and X_1, X_2 are i.i.d Bernoulli with success probability $0 < p < 1$. Then

$$\begin{aligned} \Pr[X_1 + X_2 \leq 0] = (1-p)^2 &< 1 - p = \Pr[4X_2 \leq 0] \\ \Pr[X_1 + X_2 \leq 1] = 1 - p^2 &> 1 - p = \Pr[4X_2 \leq 1] \end{aligned}$$

For a stochastic order relation to hold, the two inequalities must be in the same direction.

Now we are ready to specify the worst-case element count vector in terms of increasing convex ordering. The pattern of

worst-case item counts is that some item counts take maximum value B while other are 0. [1] Let c^* be the vector where $c_i = B, 1 \leq i \leq L/B$ and $c_i = 0$ otherwise.

Corollary 7: $X_c \leq_{icx} X_{c^*}$, and consequently

$$\Pr[X_c > A] \leq \min_{\theta>0} \frac{\mathrm{E}[e^{\theta X_{c^*}/B^2}]}{e^{\theta A/B^2}}.$$

Proof: It is easy to see that $c \leq_M c^*$. Applying Theorem 6 to the i.i.d Bernoulli random variables $\{X_i\}$, the convex function $g(x) = x^2$, we get $X_c \leq_{icx} X_{c^*}$. [2] Since $f(x) = e^{x\theta}$ is an increasing convex function of x, by definition we get $\mathrm{E}[e^{\theta X_c}] \leq \mathrm{E}[e^{\theta X_{c^*}}]$. From our earlier discussion on the Chernoff method we get

$$\begin{aligned} \Pr[X_c > A] &\leq \min_{\theta>0} \frac{\mathrm{E}[e^{\theta X_c}]}{e^{\theta A}} \leq \min_{\theta>0} \frac{\mathrm{E}[e^{\theta X_{c^*}}]}{e^{\theta A}} \\ &= \min_{\theta>0} \frac{\mathrm{E}[e^{\theta X_{c^*}/B^2}]}{e^{\theta A/B^2}}. \end{aligned}$$

For the last step we replaced θ with θ/B^2. ∎

Note that X_{c^*}/B^2 is a sum of i.i.d Bernoulli random variables, so the bound in the above corollary is exactly the Chernoff bound for the sum of L/B i.i.d. Bernoulli random variables with probability $1/g$ exceeding A/B^2. If we pick $g = \beta\frac{B^2}{A}\frac{L}{B} = \lambda^2\beta\frac{1+\epsilon}{1-\epsilon}\frac{L}{B}$, where $\beta > 1$ is a constant we can choose, then from Theorem 3 the above bound can be relaxed to:

$$\delta' \equiv \left(\frac{e^{(1-1/\beta)}}{\beta}\right)^{\frac{(1-\epsilon)}{(1+\epsilon)\lambda^2}}. \tag{3}$$

We note that δ' can be decreased by either increasing β, or decreasing ϵ, or both. If λ is small we can pick $\epsilon = 1/2$ and $\beta = 2$, and the number of groups simplifies to $g = 6\lambda^2 L/B$. If $L = nB$, then g is the same as in the previous section. However for real data we usually have $L \ll nB$, so we get a much smaller g, thus much less communication cost than if we didn't know L.

If λ is close to 1, we need to pick larger β and smaller ϵ to keep the bound small, which translates to larger communication cost. We want to have $\lambda^2\frac{1+\epsilon}{1-\epsilon} < 1$, i.e., $\epsilon < \frac{1-\lambda^2}{1+\lambda^2}$, so that a single element of size λT will not become false positive with probability at least $1 - \delta$.

Numerical example: If $\lambda = 0.1$, we can pick $\epsilon = 1/2$ and $\beta = 2$ in Equation 3, so that the false positive rate $\delta' = 0.0016$. If, say, $L/B = 0.001n$ then $g = 6n/100000$, giving a 1000-fold improvement from the analysis in Section IV. If $\lambda = 1/3$, we can pick $\epsilon = 1/2$ and $\beta = 9$, so that $\delta' = 0.0197$. Again taking $L/B = 0.001n$, we get $g = 3n/1000$. Note that the analysis in Section IV breaks down for $\lambda = 1/3$, since it gives $\delta' = 0.56$ and $g = 2n/3$. We need $g \ll n$ to achieve communication cost savings.

[1] For simplicity of computation we round L up to multiples of B. It is simple to prove that the increasing convex ordering still holds.

[2] If we were to consider other frequency moments $F_p, p > 1$, the same convex ordering results apply.

A. Estimating Iceberg Size

We can have the same size estimator as in Section IV-C. We omit the proof for the the following theorem as it is similar to the one for Theorem 4.

Theorem 8: With probability at least $1 - \delta - \delta'$ we can estimate the true size of an iceberg of size S by $\hat{S}$ with the guarantee that $S/\sqrt{2} \leq \hat{S} \leq S\sqrt{2}$.

We also can use the same unbiased estimator as in Section IV-C.

VI. Properties

In this section we discuss why we expect our algorithm to perform well on real-world data, as well as several desirable properties that it has.

A. Discussion of the Gap

All our analyses thus far have assumed that there is a gap in the data. However, our algorithm still works even when there is no such large gap, and the gap is necessary only for the purposes of providing guarantees on the performance. In practice, our algorithm performs well even when there is a small gap in the data and when the magnitude of the gap is not known.

We may also loosen the gap assumption by permitting a few elements to appear within the gap. For real problems, this is very often the case—the data usually has a long, thin tail and there are very few elements that come close to the desired threshold. Note that the false negative rate of our scheme is unaffected by such elements. Larger non-icebergs will only increase the F_2 of a group and never allow an iceberg to be missed since we keep the detection threshold fixed. Hence, the only penalty that we pay is a higher false positive rate, which only results in a slightly higher communication cost to drill down a group. This additional cost is, at worst, proportional to the number of elements in the gap. In the case of real data, this is an exceedingly small number (e.g., one or two), and hence barely affects our performance.

B. Streamed Data

When aggregating large volumes of data (e.g., Internet IP packet data), it is necessary to employ streaming algorithms to summarize the data succinctly in a single pass. Our algorithm is capable of doing this since it is already based on very light-weight sketches. In particular, each update performed locally at a node can be performed using only $O(\log(1/\delta)/\epsilon^2)$ hash operations and additions since only the sketch of a single group has to be modified for each update in the stream. Since ϵ and δ are small constants, this is essentially a constant-time update, independent of the size of the stream. We find in practice that as few as 5 to 10 estimators may suffice for each sketch.

The memory cost of our approach can be quantified by the product of the number of groups times the number of estimators for each group. In the previous section we gave some tight bounds on how many groups may be required. However, in practice, the number of groups necessary may be much smaller since our bounds assume adversarial (i.e.,

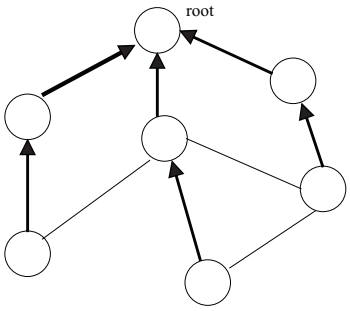

Fig. 3. The sketches can be aggregated on any connected topology. The edges indicate communication links and the heavy edges are the spanning tree along which the sketches are aggregated.

worst-case) data. Please refer to Section VII for more of these details.

C. Application to Arbitrary Topologies

Our solution can be implemented on any arbitrary connected topology due to the summable property of the sketches. Consider any communication graph G. It is possible to choose a spanning tree that is rooted at the node at which we would like to perform the iceberg detection. The protocol is then for every node in the tree to send its sketches to its parents. The parents can then sum these sketches (since all of them use the same hash functions) and pass them along to their parents. Finally the root of the tree can perform the iceberg detection as described in Section III. See Figure 3.

The communication cost for each non-root node is identical: they all have to send the same number of sketches to their parents. This means that this solution is completely unaffected by the volume of data at each node. Since the sketch sizes are independent of the number of elements inserted into them, every node has the same, succinct set of sketches. Lastly, it should be clear that it does not matter in which order the sketches are summed since the sum operation, which is simply vector addition, is commutative and associative.

VII. Empirical Evaluation

In this section we evaluate our methodology of using F_2 sketches for detecting icebergs in distributed data. We start by fine-tuning the tug-of-war sketch for our purposes. We then evaluate the performance of our algorithm on real network data by varying various parameters. We show that using our proven guarantees we can use as little as 7.5% of the space of the naive algorithm and that using 1% suffices in practice.

A. A Few Words About Sketch Size

For the tug-of-war F_2 sketch, we have an (ϵ, δ) guarantee with $32\log(1/\delta)/\epsilon^2$ or $2/(\delta\epsilon^2)$ counters. For $\epsilon = 0.5, \delta = 0.02$ this translates to 400 counters. However, this theoretical bound is quite loose. We experimented with the tug-of-war sketch on a variety of data, artificial and real, and found in all cases that a sketch of only 50 counters satisfies the $(0.5, 0.02)$ bound, i.e. it gives estimation with less than 50% relative error for more than 98% of the time.

In the experiments with our iceberg detection algorithm, we need much fewer counters. We found that 10 counter per sketch performed very well. This is due to the following reasons.

For false negative rate: We are only concerned with groups that happen to contain an iceberg. The group size has been chosen in such a way that with high probability one element (the iceberg) dominates the F_2 of the rest of the elements. We found that the tug-of-war sketch performs extremely well for such datasets, so we only need a small number of counters. (In the case that there are two icebergs in the group, the sketch will need to have at least 75% negative error to cause a false negative, which turned out to be also very unlikely.)

For false positive rate: Two factors could contribute to a false positive—the F_2 of the element counts in the group could be large and the sketch could have a large positive error. For most of the time the F_2 of the element counts is very small compared to the iceberg, and the sketch error with very high probability is not large enough to cause false positive. In other words, the deviation of F_2 plays a bigger role than the deviation of the sketch in causing false positives. Therefore a small sketch with large error still works in practice.

B. Experiments with Network Data

We tested our proposed algorithms on data collected from the Abilene network [12]. We used the destination IP addresses as the element labels. Our trace aggregated packets across several sites over a one day period. In order to simulate a large number of nodes, we distributed the packets to 100 nodes by hashing the source IP addresses uniformly at random to 100 bins. The total raw data size over the 100 nodes is 11.87M (with 4 bytes for label and 4 bytes for flow size). There are in total 140,275 unique destination addresses in this dataset. We set the bound $B = 500,000$ and the iceberg threshold $T = 1,500,000$, therefore $\lambda = 1/3$. There are two element counts between the bound and the threshold, and one element count above the threshold at $1,784,420$. Total F_1 is 3.673×10^7.

Using only the gap assumption we get the number of groups to be $g = 93516$. Adding the F_1 information we get $g = 221$, choosing $\epsilon = 1/2, \beta = 9$ so that (3) is less than 0.02. The communication cost for all the sketches is $0.884M$, counting 4 bytes per counter. The ratio to raw data is 7.5%. We encounter no false negatives or false positives at this setting. Encouraged by this, we pushed the parameters to extreme values to examine the performance of our algorithm on this dataset.

In Figure 4 we reduce the number of groups. The false negative remained at 0. We can see that false positives do not occur when $g = 60$, which corresponds to communication cost of only 2%. Even at $g = 20$ the false positive rate is still very low.

Assume that we underestimated the bound to be $B = 400,000$ instead of $B = 500,000$. So $\lambda = 1/3.75$ and we choose $\beta = 5$ in (3). Further assume that we severely underestimated F_1 to be 2×10^7 instead of 3.673×10^7. We

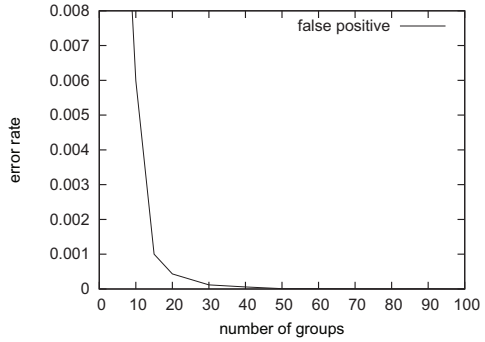

Fig. 4. Varying Number of Groups

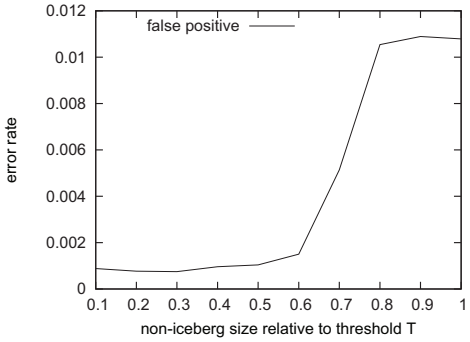

Fig. 6. Varying Non-iceberg Size

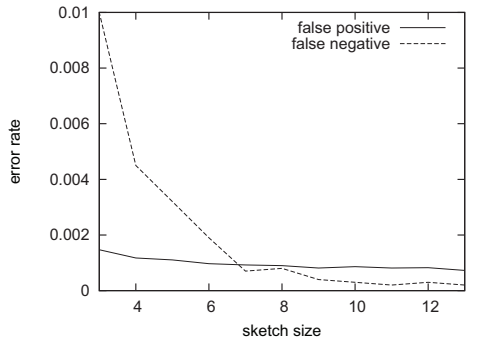

Fig. 5. Varying Sketch Size

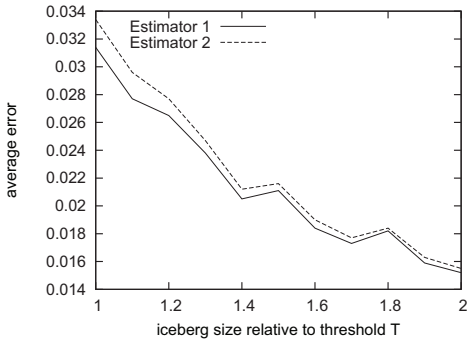

Fig. 7. Size Estimator

will get $g = 53$ which still give very good performance. This shows that it is not crucial for us to get accurate estimates of bound B or total F_1 for deriving the number of groups.

In the following we make the problem harder by reducing the iceberg threshold to $T = 1,000,000$, i.e. $\lambda = 0.5$. We also replace the large iceberg by one right at the threshold, i.e. with size $1,000,000$. We fix $g = 100$ and study how other parameters affect performance.

In Figure 5 we vary the sketch size, i.e. number of counters per sketch. We can see that false negative rate increases as sketch size decreases, which is expected. We see that false positive rate is not very sensitive to sketch size, verifying our remark about sketch size and false positive rate in the previous section.

Next we study how the size of a non-iceberg element affects the false positive rate. We insert non-iceberg of various sizes into the data. In Figure 6, the x-axis ratio is relative to the threshold T. We see that after the element reaches a certain size it starts to increase false positive rate, then it reaches a plateau where the group containing this element is very likely to report positive. We have remarked before that when a non-iceberg is close to the threshold it is hard to distinguish it from an iceberg.

Figure 7 plots the average relative error for the two size estimators when the iceberg size changes. Estimator 1 is the simple estimator, and Estimator 2 removes the bias. The x-axis ratio is relative to the threshold T. The peculiar result is that although estimator 2 is unbiased, estimator 1 has slightly less average relative error in this case.

Next we will change λ and see how it affects performance. We still use $g = 100$ and sketch size 10. We remove the 3 elements above the bound, and for each λ we set the threshold and insert an iceberg at $1/\lambda$ times the bound. Figure 8 shows the result. We see that even for higher λs (e.g., $\lambda = 0.7$, where the iceberg is less than 1.5 times the bound) the algorithm still performs well. For λ closer to 1 we will need larger g to control false positive and larger sketch size to control false negative.

Hence, we see that for real data our algorithms greatly out-perform the provided theoretical guarantees. The reason for this is that all the guarantees we give are worst-case, whereas real network data follows a highly skewed power-law distribution.

VIII. Background and Related Work

In this section we briefly survey the previous work on the issue of detecting distributed icebergs. The term *iceberg* was introduced by Fang *et al.* [13]. The term "iceberg" for a distributed heavy-hitter comes from the idea that, like icebergs in an ocean, only the tip of an item with gigantic mass can be observed from a single location. Iceberg queries are known to be useful for various applications, including detection of attacks [14], discovery of heavy-hitters in Content Delivery Networks [15], discovery of worms and other anomalies [16],

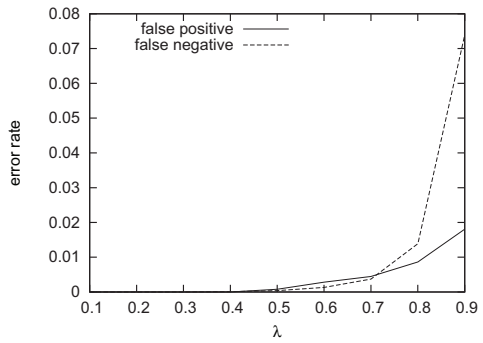

Fig. 8. Varying λ

and ensuring SLA compliance [17].

Manjhi *et al.* [6] studied the problem of discovering icebergs in a distributed environment when the nodes are in a multi-level tree topology. Their work differs from ours in that they aim to detect *recently* frequent elements, whereas we consider the problem of detection in a fixed interval. Also, our solution aims solely to detect icebergs, which allows us to discard the identities of the elements when aggregating the streams.

There also has been some work that studies a variation of the problem in which only the k most frequent items are of interest [18], [19]. Babcock *et al.* [18] studied this "Top-k" query problem, and their results were extended by Olston *et al.* [19] to support sum and average queries. Their solution has the feature that they assume that an iceberg must appear at some local node with high frequecy.

In [5], Zhao *et al.* proposed algorithms for detecting icebergs in distributed data via size-based sampling and summarization of local frequencies using a combination of quantization and Bloom filters. In their analysis, they parameterize their algorithms to give error bounds that are independent of the manner in which the iceberg is split among the local nodes.

Cormode *et al.* [20] recently proposed the problem of functional monitoring, in which local nodes continuously send updates to the central server. The goal is to minimize the amount of information sent by these nodes while still maintaining some global guarantee (e.g., detecting icebergs with high probability). This is a continuous monitoring solution and is hence incomparable with our work.

An important characteristic of our solution is that, no matter how the iceberg is split among the local nodes, the quality of our solution remains unchanged. Whereas [5] designed their scheme to attain the worst-case performance for every distribution of the iceberg across the local nodes, we automatically guarantee the same just by using the summable sketch methodology. In fact, our solution is independent of *any* characteristic of the data other than the aggregate frequency distribution, making our algorithm robust to hidden icebergs.

IX. Conclusion

In this paper we introduced the idea of using summable sketches to solve the global iceberg problem. We show that these sketches are ideal for aggregating distributed data since their behavior is independent of how the data is split. Our solution works when the data is only available as a stream at the distributed nodes, and even when the distributed nodes are organized in any arbitrary topology.

Our methodology of using of summable sketches for distributed aggregation raises the possibility of considerable future work. For example, summable sketches could be used for any frequency query on distributed data. We hope to find even more applications for this technique in the future.

Acknowledgment

This work is supported in part by NSF grants CNS-0905169 and CNS-0904743, funded under the American Recovery and Reinvestment Act of 2009 (Public Law 111-5), and NSF grants CNS-0716423 and CCF-0958490.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, 2008.
[2] Apache, "Apache hadoop," http://hadoop.apache.org/.
[3] A. Chakrabarti, S. Khot, and X. Sun, "Near-optimal lower bounds on the multi-party communication complexity of set disjointness," in *Proceedings of IEEE Conference on Computational Complexity (CCC)*, 2003.
[4] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 1996.
[5] Q. Zhao, M. Ogihara, H. Wang, and J. Xu, "Finding global icebergs over distributed data sets," in *Proceedings of the Symposium on Principles of Database Systems (PODS)*, 2006.
[6] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. opher Olston, "Finding (recently) frequent items in distributed data streams," in *Proceedings of International Conference on Data Engineering (ICDE)*, 2005.
[7] P. Indyk, "Stable distributions, pseudorandom generators, embeddings, and data stream computation," *Journal of the ACM*, vol. 53, no. 3, pp. 307–323, 2006.
[8] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar, "An information statistics approach to data stream and communication complexity," in *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, Washington, DC, USA, 2002, pp. 209–218.
[9] A. W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications*. Academic Press, 1979.
[10] A. Muller and D. Stoyan, *Comparison Methods for Stochastic Models and Risks*. Wiley, 2002.
[11] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1970.
[12] "Internet2 abilene network," http://abilene.internet2.edu/.
[13] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman, "Computing iceberg queries efficiently," in *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1998.
[14] P. Ayres, H. Sun, H. Chao, and W. Lau, "ALPi: A DDoS defense system for high-speed networks," *IEEE Journal on Selected Areas in Communications*, vol. 24(10), pp. 1864–1876, 2006.
[15] "Akamai technologies inc." http://www.akamai.com/.
[16] S. G. Cheetancheri, J. M. Agosta, D. H. Dash, K. N. Levitt, J. Rowe, and E. M. Schooler, "A distributed host-based worm detection system," in *Proceedings of the ACM SIGCOMM Workshop on Large-scale Attack Defense (LSAD)*, 2006.
[17] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Accurate and efficient sla compliance monitoring," in *Proceedings of ACM SIGCOMM*, 2007.
[18] B. Babcock and C. Olston, "Distributed top-k monitoring," in *Proceedings of ACM SIGMOD*, 2003.
[19] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proceedings of ACM SIGMOD*, 2003.
[20] G. Cormode, S. Muthukrishnan, and K. Yi, "Algorithms for distributed functional monitoring," in *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.