Mathematical Analysis

November 8, 2023

1 Mathematical Analysis

In this section, we analyze the expected AAE for the frequency estimation of BM. Then we define the "loading rate" to represent the load status of BM at a certain time.

1.1 Average absolute error (AAE)

The calculation formula of AAE is defined as $\frac{1}{|\Psi|}\sum_{(e_i \in \Psi)} |f_i - \tilde{f}_i|$, where f_i is the real frequency of item e_i, $\tilde{f}_i$ is the estimated frequency, and Ψ is the query set.

Suppose that BitMatcher records a data stream S with N items, and E is the item set ($|E| = n$). Let e_i be the i^{th} item in E, whose actual frequency is f_i. Assume that there are B entries in each bucket of BitMatcher on average, and use $fp()$ to refer to the $fingerprint()$ function. The final frequency estimation of item e_i is

$$\hat{f}_i = f_i - X_i + Y_i \tag{1}$$

where X_i is the decrement brought by the replacement strategy (when an item cannot find a vacancy in the bucket, it will decrement the smallest entry by 1, which will only lead to underestimation). And Y_i is the increment due to fingerprint collisions (which only leads to overestimation). The two error bounds of BitMatcher–the lower bound and the upper bound–result from X_i and Y_i respectively. We will calculate them separately.

1.1.1 For $E(X_i)$

We assume that the replacement strategy is "direct decrease by 1", and no fingerprint collision occurs at this time, which means $Y_i = 0$. Since there are a total of N items in the data stream, and only $2w$ buckets are used to accommodate them, on average, each bucket will have $\frac{N}{2w}$ items inserted. And since there are B entries in each bucket on average to accommodate B types of items, and the frequency of these items increase sequentially in the bucket ($entry_1.cnt \leq ... \leq entry_B.cnt$), so the maximum number of decays for item e_i occurs in this case: $entry_2 \sim entry_B$ are already occupied by other items, e_i is always in $entry_1$, and all the items coming in later are used to decrease e_i. So it follows that: $E(X_i) \leq max\ \#\ of\ decays = \min(f_i, \frac{N}{2w} \times \frac{1}{B+1})$.

1.1.2 For $E(Y_i)$

Assuming that e_i is already in $\mathcal{A}_1[\tilde{h}_1(e_i)]$ (same as $\mathcal{A}_2[\tilde{h}_2(e_i)]$). we introduce indicator variables $I_{i,j}, (1 \leq i, j \leq n)$ as:

$$I_{i,j} = \begin{cases} 1, & \begin{array}{l}(i \neq j) \wedge (fp(e_i) = fp(e_j)) \wedge \\ [\tilde{h}_1(e_i) = \tilde{h}_1(e_j) \vee \\ \tilde{h}_2(e_i) = \tilde{h}_2(e_j)]\end{array} \\ 0, & else \end{cases} \tag{2}$$

The last two conditions in the brackets ensure that e_i and e_j are mapped to at least one same bucket, and the probability can be written as

$$\begin{aligned}
&Pr[\tilde{h}_1(e_i) = \tilde{h}_1(e_j) \vee \tilde{h}_2(e_i) = \tilde{h}_2(e_j))] \\
&= Pr[\tilde{h}_1(e_i) = \tilde{h}_1(e_j)] + Pr[\tilde{h}_2(e_i) = \tilde{h}_2(e_j))] \\
&\quad - Pr[\tilde{h}_1(e_i) = \tilde{h}_1(e_j) \wedge \tilde{h}_2(e_i) = \tilde{h}_2(e_j))] \\
&\leq \frac{1}{w} + \frac{1}{w} - \frac{1}{w^2} < \frac{2}{w}
\end{aligned} \tag{3}$$

Here we assume $hash(e_i)$ is much larger than w. We have:

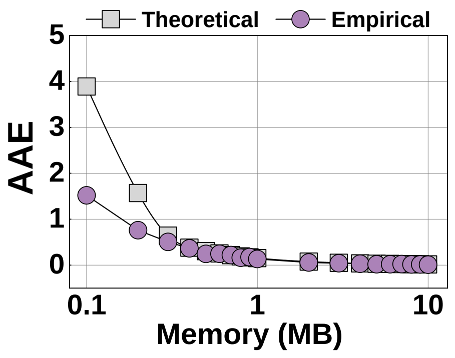
$$\begin{aligned}
E(I_{i,j}) &< Pr[fp(e_i) = fp(e_j)] \times \frac{2}{w} \\
&\leq \frac{1}{range(fingerprint)} \times \frac{2}{w} = \frac{1}{w2^{\mathcal{F}-1}}
\end{aligned} \tag{4}$$

Here we assume the length of a fingerprint is $\mathcal{F}$. And since given the two buckets mapped by e_i and e_j (and they have a shared bucket $\mathcal{A}$), e_i has a probability of $\frac{1}{2}$ being mapped into $\mathcal{A}$. So

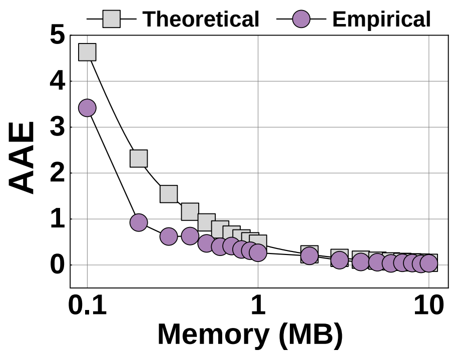
$$E(Y_i) = E(\sum_{j=1}^{n} \frac{1}{2} I_{i,j} f_j) \approx \frac{1}{2} \sum_{j=1}^{n} f_j E(I_{i,j}) = \frac{N}{w2^{\mathcal{F}}} \tag{5}$$

1.1.3 Experimental Results

Next, we compare the theoretical and empirical AAE. We note that according to the definition of AAE and Eq. (1), we have AAE $= E(|\hat{f}_i - f_i|) = E(|Y_i - X_i|)$. For cold items, their $E(X_i) \leq \min(f1, \frac{N}{2w} \times \frac{1}{B+1}) = f_i$ is a very small value. For hot items, the upper bound of $E(X_i)$ is actually too loose, because when a hot item enters a bucket, it will quickly grow into a larger entry. It won't be affected by too many replacement strategies in the process. So $E(X_i)$ of a hot item is much smaller than $\min(f1, \frac{N}{2w} \times \frac{1}{B+1})$. So to sum up, $E(X_i)$ is always a negligibly small value, and we can only consider the effect of Y_i in the experiment, that is, AAE $\lesssim E(|Y_i|)$.



(a) CAIDA



(b) IMC

Figure 1: Empirical and theoretical AAE.

We use the CAIDA and IMC datasets. For detailed information, please refer to Section 5.1.2. The experimental results are shown in Fig. 1. The abscissa is memory; the range is $0.1 \sim 1$ MB (interval is 0.1) and $1 \sim 10$ MB (interval is 1). The purple circle is the empirical AAE, and the gray square is the theoretical expectation of AAE. The fingerprint length $\mathcal{F}$ is 8 bits. We find that the theoretical values fit pretty well.

1.1.4 Space and Time Complexity

Combining Eq. 1 and Eq. 5, $\forall \epsilon > 0$, we have

$$\begin{aligned} Pr[\hat{f}_i - f_i \geq \epsilon N] &= Pr[Y_i - X_i \geq \epsilon N] \\ &\leq Pr[Y_i \geq \epsilon N] \leq \frac{E(Y_i)}{\epsilon N} \\ &= \frac{1}{w\epsilon 2^{\mathcal{F}}} \end{aligned} \tag{6}$$

Assuming that the above error probability is δ, we can get $w = \frac{1}{\delta\epsilon 2^{\mathcal{F}}}$. Since the depth of BitMatcher is 2, the space complexity is $O(2 \times \frac{1}{\delta\epsilon 2^{\mathcal{F}}}) = O(\frac{1}{\delta\epsilon 2^{\mathcal{F}}})$. In the absence of kickout operations, each insertion or query of an item only needs to check all entries in the corresponding two buckets, so the time complexity of both insertion and query is $O(1)$. We can obtain the following time and space complexity in Table 1. Notice that the d and w are the depth and width.

Sketch	d	w	Space	Insert	Query
CM[1]	$\log\frac{1}{\delta}$	$\frac{2}{\epsilon}$	$O(\frac{1}{\epsilon}\log\frac{1}{\delta})$	$O(\log\frac{1}{\delta})$	$O(\log\frac{1}{\delta})$
NI[2]	$\log\frac{1}{\delta}$	$O(\frac{1}{\epsilon^2 p} + \frac{\sqrt{\log\frac{1}{\delta}}}{\epsilon^2 p^{1.5}\sqrt{m}})$	$O(\frac{\log\frac{1}{\delta}}{\epsilon^2 p} + \frac{\log^{1.5}\frac{1}{\delta}}{\epsilon^2 p^{1.5}\sqrt{m}})$	$O(p\log\frac{1}{\delta})$	$O(\log\frac{1}{\delta})$
DHS[3]	1	$\sum_{k=1}^{3}\frac{\lambda_k}{\delta\epsilon 2^{l_k}}$	$O(\sum_{k=1}^{3}\frac{\lambda_k}{\delta\epsilon 2^{l_k}})$	$O(1)$	$O(1)$
BitMatcher	2	$\frac{1}{\delta\epsilon 2^{\mathcal{F}}}$	$O(\frac{1}{\delta\epsilon 2^{\mathcal{F}}})$	$O(1)$	$O(1)$

Table 1: Comparison of BitMatcher with SOTA.

Since many of the algorithms BitMatcher compared do not have corresponding mathematical analysis given in their original paper, we only select some of them to make a table as above. Note that some algorithms in the above table have self-contained parameters, so please go to the corresponding original paper if necessary.

1.2 Loading rate

First, we introduce the definition of the loading rate.

Definition 1.1. *If the BitMatcher contains $2w$ buckets, of which x buckets are full (no empty entries), then the* ***loading rate*** *at this time is $\frac{x}{2w}$.*

Then we have the following theorem:

Theorem 1.1. *Assuming that the BitMatcher has $2w$ buckets, each bucket has an average of B entries, each array has 1 candidate bucket, and n types of items arrive at this time. Then for the loading rate LR at this time, we have:*

$$LR \geq 1 - \left(\sum_{i=0}^{\lceil B \rceil - 1} \frac{\left(\frac{n}{2w}\right)^i}{i!} \right) e^{-\frac{n}{2w}} \tag{7}$$

Proof. We connect BM's $array_1$ after $array_2$. There are $2k$ hash functions in total at this time (the range becomes $0 \sim (2w-1)$, and the insert operation is the same as before). We define the loading rate in this case as LR'. We first prove that $LR \geq LR'$.

We consider the following: when BitMatcher processes a data stream, if one hash method is easier to find empty buckets (with at least 1 empty entry in it) than another (assuming the total number of buckets is the same), then the loading rate will increase faster at this time. Because if the hash function maps to a full bucket, it will only perform the replacement strategy without any contribution to the loading rate. So we only need to compare the probability of finding empty buckets between these two hashing methods to know which one has a higher loading rate.

We define the probability that the original BitMatcher and the connected BitMatcher find an empty bucket at a certain moment as P_{ori}^{LR} and P_{new}^{LR}. Assume that there are x_1 full buckets in $array_1$ and x_2 full buckets in $array_2$ at this time. We have:

$$\begin{aligned}
P_{ori}^{LR} &= 1 - P(all\ 2k\ mapped\ buckets\ are\ full) \\
&= 1 - (\frac{x_1}{w})^k \times (\frac{x_2}{w})^k \\
&= 1 - \frac{(x_1 x_2)^k}{w^{2k}} \qquad (8) \\
P_{new}^{LR} &= 1 - P(all\ 2k\ mapped\ buckets\ are\ full) \\
&= 1 - (\frac{x_1 + x_2}{2w})^{2k} \\
&= 1 - \frac{(\frac{x_1+x_2}{2})^{2k}}{w^{2k}} \qquad (9)
\end{aligned}$$

Since $\frac{x_1+x_2}{2} \geq (x_1x_2)^{\frac{1}{2}}$, we have $P_{ori}^{LR} \geq P_{new}^{LR}$, so $LR \geq LR'$.

Next, we calculate LR'. Note that the loading rate at a certain moment (n types of items have arrived) is actually the probability that a bucket is full at this time, which is $1 - Pr(it\ has\ a\ vacancy)$. Assuming there is no fingerprint collision at this time. There are total n types of items, and each item is randomly mapped to a certain bucket by a hash function. Given an arbitrary bucket and an arbitrary item, the probability that the flow is mapped to the bucket is $\frac{1}{2w}$. Therefore, for any bucket, the number of item types that are mapped to the bucket Z follows a Binomial distribution $B(n, \frac{1}{2w})$. In real-world scenarios, n is generally large (e.g., $n > 100$), and $\frac{1}{2w}$ is a small probability, then Z approximately follows a Poisson distribution $\pi(\frac{n}{2w})$, i.e.,

$$Pr\{Z = i\} = e^{-\frac{n}{2w}} \frac{\left(\frac{n}{2w}\right)^i}{i!} \qquad (10)$$

Suppose there are B entries in each bucket on average. The bucket will be full $\iff Z \geq \lceil B \rceil$ for this bucket. Therefore, we have

$$\begin{aligned}
LR' &= 1 - Pr(it\ has\ a\ vacancy) \\
&= 1 - Pr\{Z = 0\} - \cdots - Pr\{Z = \lceil B \rceil - 1\} \\
&= 1 - \left(\sum_{i=0}^{\lceil B \rceil - 1} \frac{\left(\frac{n}{2w}\right)^i}{i!} \right) e^{-\frac{n}{2w}} \qquad (11)
\end{aligned}$$

Combining $LR \geq LR'$, Theorem 1.1 holds. □

Theorem 1.2. *Under the conditions of Theorem 1.1, and we consider the fingerprint effect. Assuming that the theoretical upper bound of the loading rate is LR_{opt}, we have:*

$$LR_{opt} = min(\frac{n}{2w \sum_{i=0}^{B-1} \frac{2^F}{2^F - i}}, 1) \qquad (12)$$

Proof. Assume that n kinds of items have arrived. At this time, in order to achieve the highest possible loading rate (that is, the more buckets are filled), we need to fill the buckets with n items one by one. In a bucket, items are loaded starting from $entry_1$. Assume that filling $entry_1 \sim entry_b$ is expected to require $E(b)$ items, then we have:

$$E(b+1) = E(b) + \sum_{i=1}^{\infty} \{i * (\frac{b}{2^F})^{i-1} * \frac{2^F - b}{2^F}\} \qquad (13)$$

The items in $\{\dots\}$ on the right side of the above expression represent the number of items expected to be used when filling $entry_{b+1}$. Assuming $p = \frac{b}{2^F}, 0 < p \leq B$, we have:

$$E(b+1) = E(b) + \sum_{i=1}^{\infty}\{i * p^{i-1} * (1-p)\} \tag{14}$$

$$= E(b) + (1-p) * \sum_{i=1}^{\infty}\{i * p^{i-1}\} \tag{15}$$

$$= E(b) + (1-p) * \frac{1}{(1-p)^2} \tag{16}$$

$$= E(b) + \frac{2^F}{2^F - b} \tag{17}$$

According to the starting term $E(1) = 1$, we have:

$$E(B) = \sum_{i=0}^{B-1} \frac{2^F}{2^F - i} \tag{18}$$

Then on average, n items can fill $\frac{n}{E(B)}$ buckets. At this time, the loading rate is LR_{opt}. We have

$$LR_{opt} = \frac{\frac{n}{E(B)}}{2w} = \frac{n}{2w\sum_{i=0}^{B-1}\frac{2^F}{2^F-i}} \tag{19}$$

Since the loading rate cannot exceed 1, Theorem 1.2 holds. □

We use the CAIDA dataset to validate our conclusions, which contains approximately $165K$ different kinds of items. As shown in Fig. 2, we show the theoretical optimal value and theoretical lower bound of the loading rate of BM, and the empirical value is recorded every $5K$ items. The experimental results validate our theory, and the empirical value is relatively close to the ideal value. This shows that BitMatcher has very few hash collisions, that is, the information of each item is well preserved.

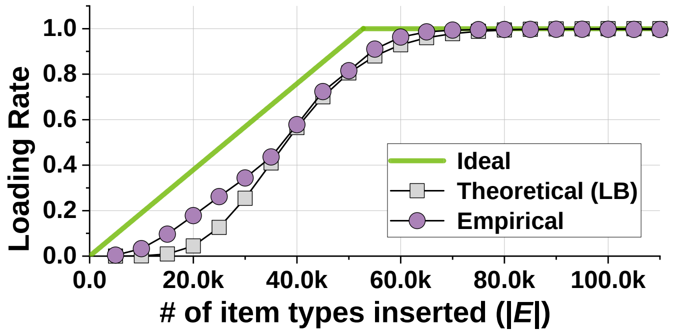


Figure 2: Empirical and theoretical LR.