Technical Report of EEM[⋆]

Yongpeng Dong, Shiyou Qian[*], Wanghua Shi, Jian Cao, Guangtao Xue

[a]*Shanghai Jiao Tong University, Dongchuan 800, Shanghai, China*

1. Introduction to WPA-REIN

1.1. Test Bench of REIN

Given the matching precision model, the next step is to characterize the relationship between the number of skipped predicates and the improved matching performance. Since the performance measurement is specific to a matching algorithm, we select REIN [1] as a test bench to quantify this relationship for two reasons. First, the index structure of REIN is predicate-oriented, which conforms to the spirit of our adjustment mechanism well. Second, predicates defined on the same attribute are indexed together in REIN, independent of other attributes. This independence makes it possible to quantify the relationship between matching time and matching precision, thus satisfying the quantification requirement.

While REIN has these advantages of being selected as the test bench, it also has one limitation. Specifically, it is only applicable to the strict scenario where the dimensionality of content space is low and events contain all the attributes. For sparse events in high-dimensional space that is formed by multiple event types, it will query and mark all buckets corresponding to attributes that are not contained in events, which will undoubtedly waste a lot of time. Therefore, to improve the applicability of REIN and Ada-REIN while enabling the adjustment performance, we augment the data structure of REIN to overcome the limitation. To ensure this paper is self-contained, a brief introduction to REIN is presented. On the basis of REIN, we augment it as a hierarchical index structure.

[⋆]This document is the results of the research project funded by the National Natural Science Foundation of China (61772334).

[*]Corresponding author.

Email address: `qshiyou@sjtu.edu.cn` (Shiyou Qian)

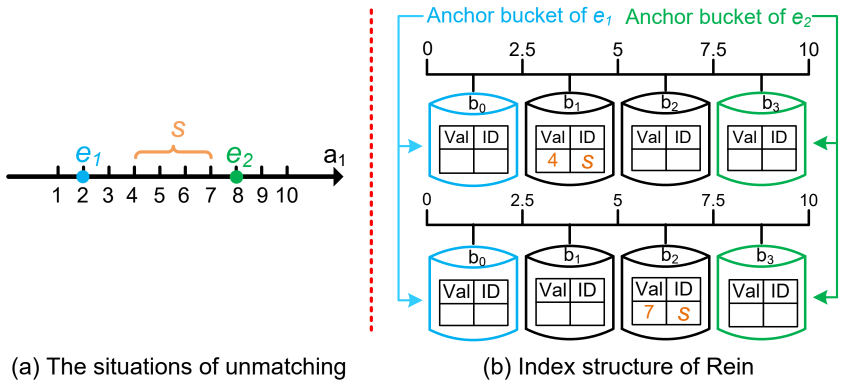


Figure 1: The searching strategy and index of REIN.

1.2. Basic Idea of REIN

The basic idea behind REIN is to quickly search unmatching subscriptions to indirectly obtain the matching ones. For each attribute, the two situations where an interval predicate does not match an event are: i) the attribute value of the event is smaller than the low value of the predicate; ii) the attribute value of the event is larger than the high value of the predicate. These two situations are depicted in Fig. 1(a) for events $e_1 : \{a_1 = 2\}$ and $e_2 : \{a_1 = 8\}$ respectively for the predicate $s : \{4 \leq a_1 \leq 7\}$.

1.3. Index Structure of REIN

Two sets of buckets are constructed for each attribute by dividing the attribute's value domain into cells and mapping the cells to buckets one-to-one. One set of buckets stores the low values of predicates and the other stores the high values, as shown in Fig. 1(b). In addition, a bit set is used to mark unmatching subscriptions.

1.4. Matching Procedure of REIN

The matching process of REIN can be divided into two stages: marking and checking. In the marking stage, for each attribute, the event value is mapped to the corresponding bucket (anchor bucket) in each of the two sets. A comparison operation is performed in the two anchor buckets, marking all unmatching subscriptions in the bit set. For the set of buckets storing the low (high) values of predicates, all buckets on the right (left) side of the

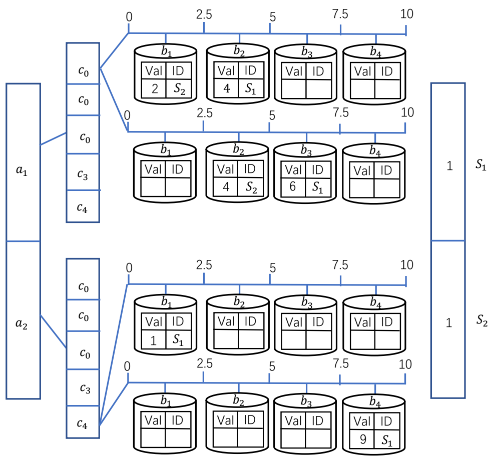


Figure 2: An example WPA-REIN indexing the subscriptions $s_1 : \{4 \leq a_1 \leq 6, 1 \leq a_2 \leq 9\}$ and $s_2 : \{2 \leq a_1 \leq 4\}$ and showing the status of matching event $e_1 : \{a_1 = 5, a_2 = 6\}$.

anchor bucket are quickly traversed to mark all unmatching subscriptions. In the checking stage, each bit in the bit set is checked, and the subscriptions represented by the unmarked bits are added to the results.

1.5. Integrating PSA into REIN

1.5.1. Test Bench of REIN

Given the matching precision model, the next step is to characterize the relationship between the number of skipped predicates and the improved matching performance. Since the performance measurement is specific to a matching algorithm, we select REIN [1] as a test bench to quantify this relationship for two reasons. First, the index structure of REIN is predicate-oriented, which conforms to the spirit of our adjustment mechanism well. Second, predicates defined on the same attribute are indexed together in REIN, independent of other attributes. This independence makes it possible to quantify the relationship between matching time and matching precision, thus satisfying the quantification requirement.

While REIN has these advantages of being selected as the test bench, it also has one limitation. Specifically, it is only applicable to the strict scenario where the dimensionality of content space is low and events contain all the attributes. For sparse events in high-dimensional space that is formed by multiple event types, it will query and mark all buckets corresponding to attributes that are not contained in events, which will undoubtedly waste a lot of time. Therefore, to improve the applicability of REIN and Ada-REIN while enabling the adjustment performance, we augment the data structure of REIN to overcome the limitation. To ensure this paper is self-contained, a brief introduction to REIN is presented in technical report. On the basis of REIN, we augment it as a hierarchical index structure.

1.5.2. Width-aware Hierarchical Index Structure

From an abstract perspective, a data structure can be viewed as a classifier that groups similar items together. Through in-depth analysis, we find that, for an interval predicate, the wider its constraint, the higher its matching probability. However, the influence of low-probability predicates on the matching result is higher than the influence of high-probability predicates. From the perspective of information entropy, the higher the entropy, the greater the uncertainty of the event. The greater the information uncertainty, the higher the information value. Therefore, small-probability predicates are more decisive on whether subscriptions are matching or unmatching.

Therefore, we augment REIN by adding an additional index level on predicate widths. Similar to REIN, the new index structure WPA-REIN also uses attributes as the first-level index feature. Then, as shown in Fig. 2, WPA-REIN uses the widths of interval predicates as the second-level index feature. To simplify the discussion, we assume that the width of interval predicates is uniformly distributed. Our goal is to classify each attribute into L classes based on the width of predicates, and then skip the predicates from the maximum width layer when performing approximate matching.

An advantage of this width-based classification is that predicates contained in the same subscription but of different widths can be handled with different priority. This matchability-aware index scheme can greatly improve matching efficiency and stability.

1.5.3. Matching Procedure of WPA-REIN

By integrating the PSA mechanism, we propose WPA-REIN, a width-aware performance adjustable variant of REIN and Ada-REIN, which is shown in Algorithm 1. Compared with REIN, an additional input parameter F is needed, which indicates the expected false positive rate of event matching. If F is set to 0, WPA-REIN runs as an exact matching algorithm. First, according to the given false positive rate F, Algorithm 1 is used to calculate the total number of skipped predicates, denoted by *skip*.

Algorithm 1 WPA-REIN

```
Require: an event $e$ and a false positive rate $F$
Ensure: $matchList$
1: Determine the skipped predicates $skip$ by Algorithm 1;
2: for $i = 1 \rightarrow m$ do
3:     if $skip[i] == False$ then
4:         Perform matching like REIN;
5: for each subscription $s$ in subscription list do
6:     if the bit of $s$ not marked in the bit set then
7:         Add $s$ to $matchList$;
```

Second, for the remaining predicates, a matching process similar to REIN is performed. Finally, check the unmarked subscription and obtain the matching results *matchList*. Otherwise, WPA-REIN works in the approximate matching mode with the specified false positive rate. The switch between exact matching and approximate matching is dynamic and seamless, without reconstructing the underlying index structure. Furthermore, different events can be matched at different false positive rates.

1.6. Matching Time Estimation Model of WPA-REIN

On the basis of the matching precision model, we further quantify the relationship between the number of skipped predicates and the rate of performance improvement based on the data structure and matching procedure of WPA-REIN.

Lemma 1. *Suppose the predicate values defined on the attributes are uniformly distributed in a set of subscriptions. For an attribute a_i with frequency $f_i(0 \leq i \leq m)$ and L levels, the average bucket size b_i on a_i, which denotes the average number of predicates in buckets, is $z_i = \frac{2Lf_i}{b(L+1)}$.*

Proof. In WPA-REIN, for the m event attributes, each attribute has an L-level index structure and b buckets. For an attribute a_i, when the predicate values defined on a_i are uniformly distributed, except for empty buckets, these predicates are evenly stored in the $\frac{b(L+1)}{2L}$ non-empty buckets, so the bucket size is $z_i = \frac{2Lf_i}{b(L+1)}$. For attributes with different frequencies, their bucket sizes are different. □

Lemma 2. *The matching cost of REIN is characterized by $\sum_{i=1}^{m} z_i(2\beta + b\gamma) + mb\delta + n\epsilon$.*

Proof. The matching process of REIN consists of the marking stage, comparing stage and checking stage. In the marking stage, predicate traverse operations in buckets and bucket switch operations are performed. In the comparing stage, for each attribute, the comparing operation occurs between the attribute value of the event and the low and high value of the predicate. In the checking stage, each bit in the bit set is checked to determine the matching subscriptions.

Given the number of buckets b and the number of attributes m in content space, let β be the unit time to compare a predicate in a bucket, γ the unit time to traverse the predicates in a bucket, δ the unit time to switch buckets when traversing, and ϵ the unit time to check a bit. When matching events, for each attribute, comparison operations are performed in two anchor buckets to find the unmatching subscriptions, thus the cost is $\sum_{i=1}^{m} z_i 2\beta$ where z_i is the average bucket size of attribute a_i. After comparison, the predicates stored in the other buckets are quickly traversed, only marking the corresponding bit in the bit set. For the low values and high values of predicates stored in the buckets, the whole b buckets will be traversed, so the cost is $\sum_{i=1}^{m} z_i b\gamma$. In addition, the cost of switching b buckets is $b\delta$ for each attribute, so the total cost is $mb\delta$. In the checking stage, each bit is checked, so the cost is $n\epsilon$. Therefore, the total matching cost of WPA-REIN can be expressed as:

$$T = \sum_{i=1}^{m} z_i(2\beta + b\gamma) + mb\delta + n\epsilon \tag{1}$$

□

Theorem 1. *Given the excepted number of skipped predicates Y and number of predicate categories L, the matching performance improvement by skipping Y predicates in WPA-REIN is*

$$R = \frac{\sum_{a_j \in \mathbb{A}_s} \frac{4L\bar{k}n\beta}{mz_j(L+1)} + \frac{bmY(L+1)}{2L\bar{k}n\delta} + \gamma Y}{\sum_{i=1}^{m} z_i(2\beta + b\gamma + bh) + mb\delta + n\epsilon}.$$

Proof. Skipping Y predicates in the matching process of WPA-REIN decreases the comparison cost, traversing cost and switching cost. Firstly, as can be seen from Algorithm 1, we preferentially select the attributes that can be fully skipped at the attribute-width granularity, denoted by set $\mathbb{A}_s$.

Then, we can easily calculate the total number of predicates as $\bar{k}n$ and the average bucket size of each attribute is $\frac{2L\bar{k}n}{mb(L+1)}$. Furthermore, we can also obtain the number of skipped buckets $\frac{bmY(L+1)}{2L\bar{k}n}$. Finally, the sum of these reduced costs is $\sum_{a_j \in \mathbb{A}_s} \frac{4L\bar{k}n\beta}{mz_j(L+1)} + \frac{bmY(L+1)}{2L\bar{k}n\delta} + \gamma Y$. The total matching cost of WPA-REIN is given in Equation (7). Therefore, the matching performance improvement R obtained by skipping Y predicates is

$$R = \frac{\sum_{a_j \in \mathbb{A}_s} \frac{4L\bar{k}n\beta}{mz_j(L+1)} + \frac{bmY(L+1)}{2L\bar{k}n\delta} + \gamma Y}{\sum_{i=1}^{m} z_i(2\beta + b\gamma + bh) + mb\delta + n\epsilon}. \tag{2}$$

□

1.7. Parallelization of WPA-REIN

With the continuous increase in the data generation rate, it is very difficult to make the serial event matching meet real-time requirements under massive data. Parallel processing has become a trend, and whether it can well support parallelization has also become an important indicator to measure the matching algorithm.

Since the buckets that events fall into are dynamically changing and the frequency of attributes varies greatly, matching task allocation at attribute granularity and bucket granularity leads to serious load imbalance and high computation cost. Therefore, benefiting from the fine-grained hierarchical index structure of WPA-REIN, it can well support attribute-width granularity parallelism mechanism, which has the advantage of good load balancing and low computational cost. Consequently, we use a thread pool to respond to matching subtasks quickly without too much process and thread switching overhead. First, we count the predicates that are not skipped at each width level of each attribute, and sort subtasks from the largest to smallest. Then, we iteratively allocate the largest subtask to the logic thread with minimum workload. At last, we sort the subtask set of each thread from the smallest to largest by attribute-width granularity, so that the subtasks of a same attribute are adjacent and can be allocated to as few threads as possible.

2. Performance of WPA-REIN

2.1. Experiment Setup

WPA-REIN is implemented in C++ language and compiled by g++ 9.3.0 with -O3 optimization enabled on an Ubuntu 20.04 system. All the experiments are conducted on an AMD R9 5900X CPU with a base clock speed of 3.7 GHz. The capacity of main memory is 64 GB.

Table 1: Parameters used in experiments

Note	**Description**	Value
n	Number of subscriptions	1M
d	Dimensionality of content space	500
α	Parameter of attributes' Zipf distribution	0
k	Subscription size	5
w	Matchability of interval predicates	0.5
b	Number of buckets in REIN	1000
L	Number of index layers in WPA-REIN	5

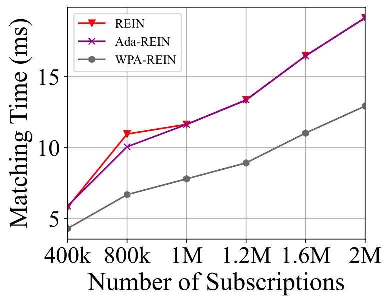
For brevity, the value domain of attributes is normalized on [0, 1.0] from which the predicate values and event values are uniformly generated with precision 10^{-6}. The parameters used in the experiments are listed in Table 1. 1000 events are matched in each experiment and each experiment is repeated 10 times. The matching time is used to evaluate the performance of the matching algorithms.

In addition to REIN, WPA-REIN is compared with Ada-REIN [1], which realizes coarse-grained performance adjustment [2]. Specifically, given the false positive rate F and the frequency of attributes, Ada-REIN computes the attributes to be skipped. Ada-REIN works based on the assumption that attributes follow the Zipf distribution. And there is no performance improvement under this assumption compared to REIN. Then, in the next experiments, we compare WPA-REIN of 0.1% false positive rate with REIN and Ada-REIN to evaluate the impact of different parameters.

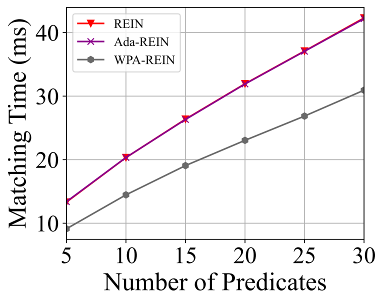
2.2. Experiments Results

Impact of the number of subscriptions n. The number of subscriptions n is a critical parameter to measure the workload, which has a significant impact on the matching time. Fig. 3(a) shows the matching time of the three tested algorithms with different settings of n. Intuitively, all algorithms take longer to match events with more subscriptions and exhibit an upward trend. WPA-REIN performs better than their counterparts in all situations. Compared with REIN and Ada-REIN, WPA-REIN reduces the matching time by 32.0% and 32.7% respectively on average. When $d = 20$, Ada-REIN has the same

[1]Ada-REIN is the original conference version published in INFOCOM 2019.



(a) Effect of n



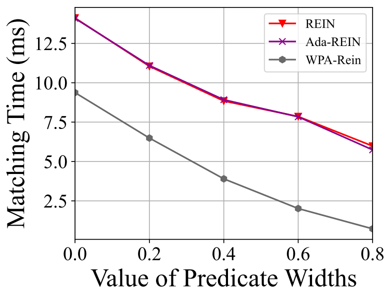
(b) Effect of k

Figure 3: Matching time with different subscription number and size.

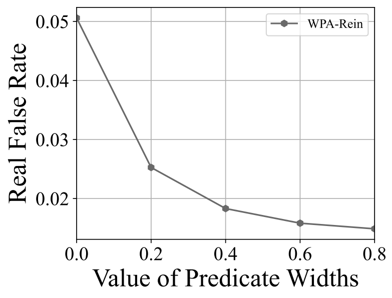
performance as REIN. This is because Ada-REIN does not skip attributes. However, WPA-REIN adopts more fine-grained skip mechanism and achieves a significant performance improvement.

Impact of subscription size k. As shown in Fig. 3(b), subscription size k has an important effect on the performance of event matching. To measure the effect of k, we set $d = 30$ and vary k from 5 to 30. When $k = 5$, WPA-REIN makes an approximate 32% performance improvement over REIN and Ada-REIN, whereas the improvement reduces to 27% when $k = 30$. The cause of this issue is due to the increased frequency of attributes. On the one hand, for such subscriptions skipping even a small number of attributes and buckets will offend the expected false positive rate, which limits the adjustment space of WPA-REIN. On the other hand, the predicates in WPA-REIN are layered according to predicate widths, effectively utilizing the expectation of false rate F.

Impact of predicate width w. The matching probability is determined by subscription size k and predicate width w. Fig. 4(a) shows how width affects the matching time by varying w from 0.2 to 1. In this set of experiments, width w is a dynamic parameter. For instance, 0.8 represents the minimum value in the range of 0.8 to 1. k is set to 5 to ensure there are always some matches. This setting makes the predicates sparsely distributed in cells. As the predicate width increases, the matching time of REIN, Ada-REIN and WPA-REIN sharply decreases. This is attributed to the backward matching strategy of the three algorithms. Furthermore, compared to REIN and Ada-REIN, the performance of WPA-REIN can be improved to 87.4% and 87.9%



(a) Effect of w



(b) Effect of w

Figure 4: Matching time with different predicate widths.

respectively. Since WPA-REIN contains a more fine-grained skip mechanism, which preferentially selects the levels with larger width, so the false positive rate can be minimized, as shown in Fig. 4(b)

Impact of the dimensionality of content space d. Supporting high dimensional spaces is one of the challenges for matching algorithms. Given n and k, the higher the dimensionality, the sparser the workload in the space. Fig. 5 shows the remarkable tendency that the three algorithms behave monotonically. This is because the three backward matching algorithms have to mark all the unmatching subscriptions on each attribute. Furthermore, when subscriptions are uniformly distributed, the performance of Ada-REIN becomes increasingly worse. Even when $d = 100$, the performance of Ada-REIN is not as good as that of REIN. Compared with REIN and Ada-REIN, on average the performance of WPA-REIN is improved by 21.7% and 24.2%, respectively.

Impact of attribute distribution α. The skewed distribution of subscriptions and events is divided into two categories. One is the value distribution and the other is the attribute distribution. Because the uneven distribution of values has little effect on the matching time, here we show the experiment results under skewed attribute distribution of both subscriptions and events in Fig. 6(a) where d is 3000. Considering the key parameter α in the Zipf distribution, the larger the α, the more serious the skew of the data, which results in some popular attributes. As shown in the figure, the performance of the three algorithms is sensitive to the distribution of attributes.

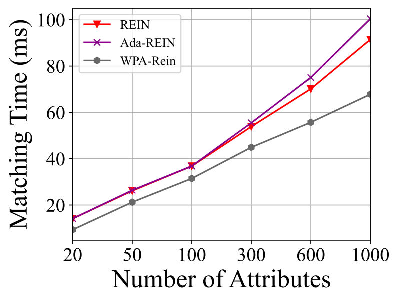


Figure 5: Matching time with different dimensionality.

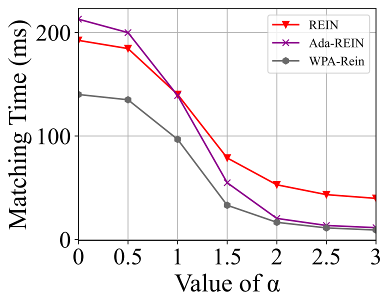
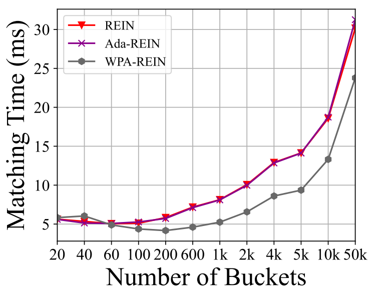
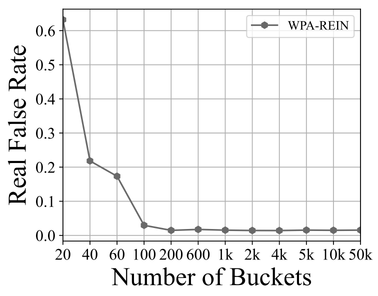


Figure 6: Matching time with different attribute skew.



(a) Effect of b

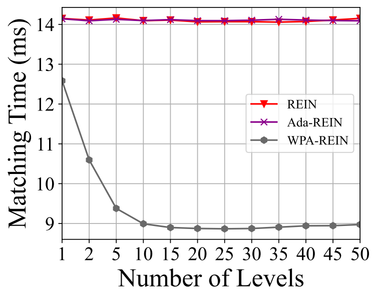


(b) Effect of b

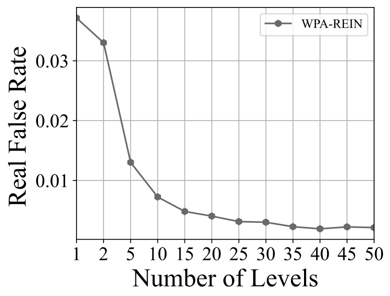
Figure 7: Matching time with different buckets.

For Ada-REIN with $\alpha = 3$ where the popular attributes are concentrated, more attributes belong to the infrequent category, which provides more candidates to skip without offending the expected false positive rate, even though, compared with REIN and Ada-REIN, WPA-REIN improves the performance by 70% and 20% respectively. On the other hand, when $\alpha = 0$, which denotes uniform distribution, the performance of Ada-REIN is not as good as that of REIN. On the contrary, WPA-REIN overcomes this disadvantage of Ada-REIN on the basis of the fine-grained skip mechanism. WPA-REIN outperforms REIN and Ada-REIN by up to 27.2% and 34.2%, respectively.

Impact of the number of buckets b. As shown in Fig. 7(a), b is a key parameter that affects the performance of WPA-REIN. When b varies from 20 to 100, WPA-REIN performs almost as well as REIN and Ada-REIN. By analyzing the matching process of REIN, Ada-REIN and WPA-REIN,



(a) Matching time



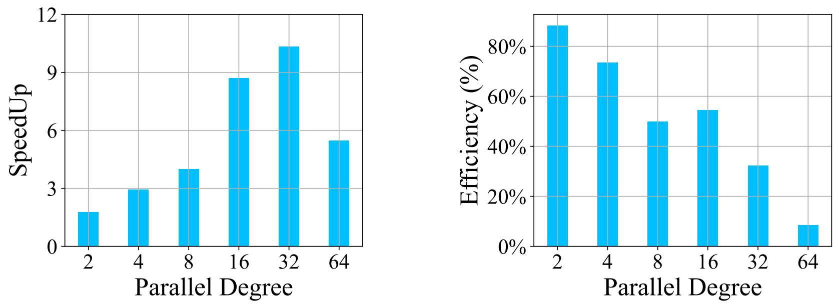
(b) Measured false positive rate

Figure 8: Matching time and measured false positive rate with different L.

comparison operations are only executed in one bucket for an attribute. Obviously, the smaller b, the larger the size of the buckets and the higher the cost of comparison. Therefore, the performance of WPA-REIN is similar to that of REIN and Ada-REIN when b varies from 20 to 100. As increasing b reduces the cost of comparison operation, when $b = 1000$, the performance of WPA-REIN is improved by at most 35.6% and 35.6% respectively compared with REIN and Ada-REIN.

Fig. 7(b) represents another crucial point, that is, the measured false rate has a sudden decline, with an increase in b. The reason is that we adopt the mechanism of skipping the minimum bucket at the attribute-width granularity, this mechanism can make the distribution of attributes more concentrated. When the number of buckets increases, the size of the bucket of the level with larger predicate width decreases rapidly, while the size of the bucket with smaller predicate width changes less. Therefore, when we choose to skip the minimum bucket, we give priority to skipping the bucket with the larger predicate width level. Furthermore, predicates in this level have a large matching probability, and the probability of false positives is relatively low.

Impact of the number of levels L. Fig. 8(a) shows the matching time of WPA-REIN under different settings of L. We observe that L has important effects on WPA-REIN. Compared with REIN and Ada-REIN, WPA-REIN improves the performance by up to 37.1% and 37.1%, respectively on average. This is due to the probabilistic hierarchical index structure based on predicate width, which changes the distribution of subscriptions in buckets. In other words, the hierarchical index structure makes subscriptions more concentrated in



(a) Speedup (b) Efficiency

Figure 9: The parallelization effect of WPA-REIN

certain buckets and increases the number of empty buckets. So more empty buckets can be skipped before matching, which reduces the cost of traversing buckets.

Fig. 8(b) shows that the real false rate dramatically drops from 3.7% to 0.21% when L increases from 1 to 50. The contributing factor for this phenomenon is the width-aware hierarchical index structure. During the process of selecting skipped buckets, priority should be given to the larger width level, to reduce the false rate.

Parallelization effect of WPA-REIN. To evaluate the parallelization effect of WPA-REIN, we measure two metrics: speedup and efficiency. The parameters are set by default. We parallelize WPA-REIN at the attribute-width granularity. The experiment results are shown in Fig. 9. We observe that when the parallel degree is 32, the speedup reaches the maximum value of 10.33, and the efficiency is as high as 95.67% when the parallel degree is 2. The main reason for the decrease in the speedup of 64 threads is that with an increase of parallelism, the matching process time continues to reduce, but thread switching takes much longer, resulting in a decrease in parallel efficiency, as shown in Fig. 9(b). Overall, WPA-REIN is able to support the fine-grained parallelism of the algorithm, achieving good parallelism when the number of threads is between 16 to 32.

3. Conclusions

Acknowledgement

This work was supported by the National Science Foundation of China (61772334).

References

[1] S. Qian, J. Cao, Y. Zhu, M. Li, REIN: A fast event matching approach for content-based publish/subscribe systems, in: IEEE INFOCOM, 2014, pp. 2058–2066.

[2] S. Qian, W. Mao, J. Cao, F. L. Mouël, M. Li, Adjusting matching algorithm to adapt to workload fluctuations in content-based publish/subscribe systems, in: IEEE INFOCOM, 2019, pp. 1936–1944.