IWApriori: An Association Rule Mining and Self-updating Method Based on Weighted Increment

Yonghua Huo
The 54th Research Institute of CETC
Shijiazhuang, Hebei, China

Jing Dong
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China

Zhongdi Ge
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China

Ping Xie
The 54th Research Institute of CETC
Shijiazhuang, Hebei, China

Na An
The 54th Research Institute of CETC
Shijiazhuang, Hebei, China

Yang Yang*
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
yyang@bupt.edu.cn

***Abstract*—The mining of association rules plays an important role in fault prediction. Many studies have shown that there is an obvious temporal and spatial correlation between the failure records of the cluster system. Therefore, most cluster system failure prediction engines are built based on causal correlation analysis between log events. However, the original system log file usually contains a large number of invalid records (duplicate or non-fault related records), which makes the mining of event correlation extremely difficult and seriously affects the efficiency and accuracy of fault prediction. Therefore, this paper proposes an association rule mining and self-updating method based on weighted increment, named IWApriori (improved weighted Apriori algorithm). The method includes two important steps: 1) log preprocessing; 2) mining and updating of association rules based on improved algorithm IWApriori. This method can effectively improve the rule completeness and realize the efficient mining and updating of rules in the whole life cycle of the system. In addition, we used the real log data set Blue Gene/L to validate our method. The results show that our association rule mining method is better than other methods in terms of time performance, space performance and the effectiveness of mining rules.**

Keywords—Association Rule Mining, Log Filtering, self - updating

I. INTRODUCTION

At present, most failure prediction engines for cluster systems are based on system logs. This is because that logs record in detail the various events which describe the state changes of clusters, so we can track system behavior more accurately in the long term. Studies show that there is obviously the time correlation and spatial correlation between the event logs [1]. However, the original system logs contain a large number of invalid records (repeated or unrelated to the failure) which may interfere with the discovery of causation, so how to use the spatio-temporal correlation of events to effectively analyze the system log and realize the mining of association rules is the focus of this paper.

The previous failure prediction method based on association rule mining has the following disadvantages: 1) The prediction result provides too little information. Most methods can only achieve the prediction of failure levels or coarse-grained failure types. But a failure event has many important attributes (such as time, node location, event type, event severity) that cannot be ignored. For example, if a failure is predicted without node and time information, the manager cannot take appropriate action. 2) The recall of prediction is generally low. In the log filtering phase, the past work usually only keeps the first record and deletes all subsequent records, which may eliminate the information that is crucial to analyze the causal relationship between events. The rule extraction phase, due to the window-size restriction, the failure rule expression is not complete enough, so that the recall rate of the failure prediction is not high. 3) The time complexity is too high. Software updating frequently in clusters, that means the generation of new events. The previous methods mines and update rules in a cold-start manner and with high time and space consumption, which are not suitable for dynamically changing production environments.

In view of the above problems, this paper proposes a new association rule mining method named IWApriori (improved weighted Apriori algorithm). Specifically, the innovations and contributions of this article are summarized as following:

- A new log preprocessing method is proposed, which can realize adaptive event identification and log filtering according to semantic similarity and time correlation of events, and retain more event information by adding fields.
- A new weighted association rule mining method (IWApriori) which has very good spatio-temporal performance is proposed to solve the problem of fixed time window's influence on transaction partition and frequent item mining of small probability events. What's more we put forward a rules-updating strategy

based on IWApriori, which can realize the efficient updating of rules in the whole life cycle of the system by self-starting. The formatter will need to create these components, incorporating the applicable criteria that follow.

The rest of this article is organized as follows. Section II reviews some related work. Section III describes our association rule mining algorithm IWApriori (improved weighted Apriori algorithm). Simulation results and corresponding discussions are presented in Section ˜. Finally, Section V summarizes this paper.

II. RELATED WORK

In the past few years, a large number of failure prediction methods have emerged. [1,2,3] combines statistical methods and machine learning models to predict failure. But this kind of method cannot provide detailed prediction information. Therefore, failure prediction method based on event association rule mining is the main focus of our research. Failure prediction results are mainly affected by two aspects: log preprocessing and association rule mining.

The main task of log preprocessing is to delete the redundant log records. A good log filter not only has a high compression rate, but also retains as much complete event correlation information as possible. The spatio-temporal filtering method (STF) proposed by Liang [2] is the most widely used for log filtering method, but has a high rate of information loss. Zheng et al. [4] solved the above problems by recording the event start-time, end-time, event count and location information, but the classification of events requires the support of expert experience. Kobayashi et al. [5] deleted events with weak correlation based on causal graph. Liang Y et al. [6] proposed an adaptive semantic filter (STF) to judge whether two events are redundant according to the semantic correlation and time interval between events. Sheng Di et al. [7] improved the ASF and redefined the calculation method of similarity between events, but many artificial weight was involved, which was not convenient for the promotion of the method.

In terms of rule mining, Zheng [8] uses association rule mining and Bayesian model to predict failure nodes in the future. But its effect on large-scale cluster system is debatable. The rule extraction in [9,10,11,12] is all based on fixing time window. The expression of failure rules is not complete enough, so the recall is low. Fu X et al. [12] and Yu Y et al. [13] proposed two types of association rule mining that are not limited by window size, and the recall is greatly improved. However, they do not meet the requirements of dynamics. When a large number of new alarm logs are generated, the above method can not be automatically mine and update the rules, and the time complexity is too large. We hope to provide a smarter and faster way to automatically acquire and update rules during the system operation, while finding a balance between accuracy and granularity of failure prediction.

III. IWAPRIORI

A. Overview

The workflow of IWApriori is shown in Fig.1, including two important steps: log preprocessing and association rule mining and updating. The main working principle is: in the log preprocessing phase. The historical log data are marked by event_ID and the unique event_ID list will be stored in the ELIB. The redundant events will be filtered based on temporal and semantic relevance. After that, the sliding time window is used to partition the transaction set, and then event rules will be mined by the improved IWApriori algorithm.

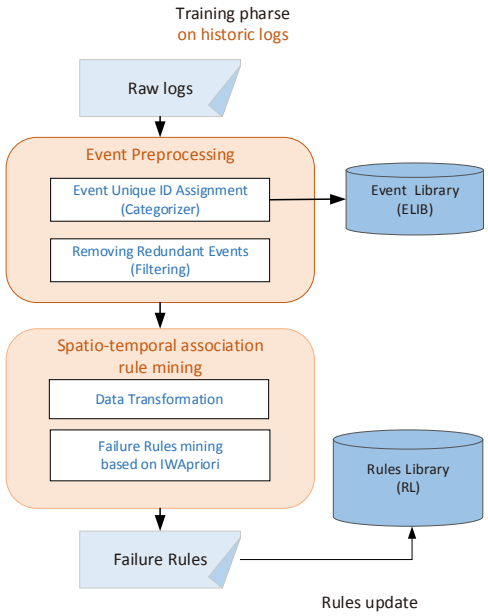

Fig. 1. Method framework

B. Log preprocessing

We proposed a new log preprocessing method. The main principle is to calculate the semantic similarity $S(e_1, e_2)$ between every event and its following events within the time threshold T_{th}, group the logs based on $S(e_1, e_2)$, assign a unique event_ID for each group, and filter out the events below the semantic threshold S_{th}. S_{th} increases with the increase of T_{th}. That means, when the time interval between events is small, even if the similarity between two events is low, it will be regarded as repeated record. When the time interval is large, only the semantic relevance is very high, it will be regarded as repeated record. For example, for Blue Gene/L system, the threshold set is as follows:

$$S_{Th} = \begin{cases} 1, & if\ T_{th} \in [20\,\text{min}, \infty) \\ 0.7, & if\ T_{th} \in [5\,\text{min}, 20\,\text{min}) \\ 0.6, & if\ T_{th} \in [5\text{s}, 5\,\text{min}) \\ -1, & if\ T_{th} \in [0, 5s) \end{cases} \quad (1)$$

The similarity between two events in the method is the weighted sum of similarity of important fields, as shown in equation (2) :

$$S(e_1, e_2) = \sum_{\alpha_i \in e_1, \beta_i \in e_2} \omega_i \times s_i(\alpha_i, \beta_i) \quad (2)$$

i is the field index. α_i, β_i is the i field of event e_1, e_2, w_i is the weight. s_i is the similarity calculation method of field i. In this paper, we select two fields: event location and description to calculate the $S(e_1, e_2)$. The weight is set as (0.4,0.6). The specific calculation method is as follows：

- Similarity of event locations: in RAS logs, the location of the event is represented by a location code. For example, R1F-M0-N14 represents a 3-layers location in a compute rack 1F: mid-plane 0: node board 14. In this case, the similarity of location is defined as

follows: $s_1(\alpha_1, \beta_1) = 0.25 * n, n$ is the number of the highest same code layers.

- Similarity of event description: The event description is usually a piece of text, but parameters such as host name, IP address, file name often appear, causing the diversity. So firstly we replace the variables with "*" to reduce the number of event_ID. Then the description field is divided into words, represented by a text set. In this paper, Jaccard coefficient is used to calculate the semantic similarity between event descriptions. At the same time, in order to retain as many fatal failure events and differential events as possible, We use the event severity and the number of rare keywords to weaken the similarity, that is, the higher the event severity , the stronger the difference, and the smaller the similarity should be. Equation (3) gives the calculation formula:

$$s_2(\alpha_2, \beta_2) = \frac{1}{k} \times \zeta \times \frac{|\alpha_2 \cap \beta_2|}{|\alpha_2 \cup \beta_2|} \tag{3}$$

k refers to the number of rare keywords(the words with low word frequency) appearing in α_i and β_i. ζ is the weight corresponding to the severity level of the event, between 0 and 1, the higher the level, the smaller it is. $\frac{|\alpha_2 \cap \beta_2|}{|\alpha_2 \cup \beta_2|}$ is the Jaccard coefficient.

It is worth noting that we refer to the ideas in [4] when filtering, add three new fields(start-time, end-time and event count) for each event so as to retain more failure propagation information.

C. Event Association Rules mining and updating

We propose a new weighted incremental association rule mining method (IWApriori). It is mainly divided into three parts: the transaction set division, association rule mining, and association rule updating.

1) transaction set division:

Transaction: is a sequence of events obtained through the time window. A transaction is denoted as $t = \{e_1, e_2, \dots, e_q\}$. The database can be represented as a collection of transactions $D = \{t_1, t_1, \dots, t_N\}$.

Event (e): is a single line of text in system logs containing multiple fields reporting changes in system status. In this paper, the event is expressed as a seven-tuple, that is:

$$e_i = \{timestamp, time, location, severity, type, keywords, event\_ID\}$$

It is assumed that $[T_s^A, T_e^A]$ and $[T_s^A, T_e^A]$ represent the start-stop time periods of Event A and Event B respectively. If $[T_s^A, T_e^A + W] \cap [T_s^B, T_e^B] \neq \emptyset$, it is considered that there may be an association between events A and B. We define a sliding time window W to divide the transaction set. The specific method is that the start time of the window W is the start-time of the first record e_1 in the event log. Take all the event records of start-time in $\left[T_e^{e_1},\ T_e^{e_1} - T_s^{e_1} + W\right]$ convert them into a transaction $t = \{e_1, e_2, \dots, e_q\}$, move the time window making the start time is the start-time value of the next record. The above process is repeated until the end time of the time window reaches the end time of the last event. Compared with the general transaction method based on a sliding time window, the above method can avoid the problem of filtering out the same alarm stream for a long time. Thereby reducing the time interval between the precursor event and the subsequent event.

2) Weighted association rule mining algorithm-IWApriori

Association rule: an association rule is the implication of the form, X is the premise of the rule, Y is the result of the rule.

$support(X \Rightarrow Y)$: The support degree of $X \Rightarrow Y$ represents the frequency of the transactions which contains both X and Y.

$confidence(X \Rightarrow Y)$: The confidence degree of a rule represents the probability of the occurrence of Y when X appears.

The classic frequent item set mining algorithm considers the importance of each item in the database to be the same and the distribution of the items in the database is uniform. Therefore, the classic association rule mining algorithm processes each item in the database in an equal and consistent manner. However, the actual situation of the database is not the case, especially in the log records. The number of fatal fault events is far less than the number of non-fatal fault events. In order to explore the causal relationship between non-fatal events and fatal events, this paper retains non-fatal events as in the literature [12, 13]. This poses a problem that the FESs containing fatal events are likely to be filtered out under the same support threshold. To this end, this paper comprehensively considers the impact of event attributes and frequency on association rules, and proposes a new method for calculating weighted support and confidence. Some important definitions involved in the method are as follows:

We set $D = \{t_1, t_1, \dots, t_N\}$ be a collection of database transactions. A transaction is denoted as $t = \{e_1, e_2, \dots, e_q\}$,representing a sequence of events. $I = \{e_1, e_2, \dots, e_M\}$ is a collection of event_ID with a total of M event type. X, Y are two k-ary ESs.

***k*-ary Event Sequence (ES):** is a sequence of k events with different event_ID in chronological order. For example, event sequence (e_1, e_2, e_3) is a 3-ary ES.

Event weights $w(e_i)$: In this paper, each event event_ID is given a weight, which is used to represent the importance of the event. It is mainly based on two properties of the event (the number of fatal events and the number of non-fatal events) and the frequency of the event. Its calculation formula is as follows:

$$w(e_i) = sigmoid\left(\frac{\gamma C_{non-fatal} + \eta C_{fatal}}{|e_i|}\right) \in (0,1) \tag{4}$$

In the formula, $C_{non-fatal}$ is the number of non-fatal events, C_{fatal} is the number of fatal events. $\gamma < \eta, \gamma + \eta = 1$, takes $\gamma = 0.2, \eta = 0$. $|e_i|$ represents the number of times e_i occurs. This represents the more serious the level of the event and the lower the probability of occurrence, the greater its importance.

Fatal event and non-fatal event: non-fatal events are generally minor error records that the system can recover by itself. Fatal events usually lead to application/system crash and other serious consequences, which is the main target of failure prediction. In this article, we consider the severity FAILURE and FAULT as fatal events.

***k*-ary ES weight $W(X)$:** The ES weight is a summary of the weight of events in the ES. It is calculated as follows:

$$W(X)=\frac{\prod_{i=1}^{k}{}_{(\forall e_i\in X)}\, w(e_i)}{\sum_{i=1}^{k}{}_{(\forall e_i\in X)}\, w(e_i)} \tag{5}$$

Weighted support. The calculation is as (6), where N is the total number of transactions and n_X is the number of transactions containing X.

$$wsupport(\mathrm{X})=W(\mathrm{X})\times\frac{n_X}{N}=\frac{n_X\prod_{i=1}^{k}{}_{(\forall e_i\in X)}\, w(e_i)}{N\sum_{i=1}^{k}{}_{(\forall e_i\in X)}\, w(e_i)} \tag{6}$$

Weighted confidence. The weighted confidence is the ratio of the weighted support that satisfies $X\cup Y$ in the transaction database to the weighted support that contains X. Its calculation method is as follows:

$$\begin{aligned} wconfidence(X\Rightarrow Y)&=\frac{wsupport(X\bigcup Y)}{wsupport(X)}\\ &=\frac{n_{(X\cup Y)}}{n_X}\bullet\frac{\prod_{i=1}^{|X\cup Y|}{}_{(\forall e_i\in X\cup Y)}\, w(e_i)}{\prod_{i=1}^{k}{}_{(\forall e_i\in X)}\, w(e_i)}\bullet\frac{\sum_{i=1}^{k}{}_{(\forall e_i\in X)}\, w(e_i)}{\sum_{i=1}^{|X\cup Y|}{}_{(\forall e_i\in X\cup Y)}\, w(e_i)} \end{aligned} \tag{7}$$

The weighted association rule-mining algorithm often does not satisfy the nature of frequent set down closure. However, the improved method in this paper can satisfy the nature of frequent set down closure when mining numerical data. The proof is given below:

Proof: Set the minimum support degree $minSup$, $Z=X\cup\{e'\}, X\subset Z$, we can get the follow equation according to the definition of weighted support.

$$wsupport(\mathrm{Z})=\frac{n_Z\prod_{i=1}^{|Z|}{}_{(\forall e_i\in Z)}\, w(e_i)}{N\sum_{i=1}^{|Z|}{}_{(\forall e_i\in Z)}\, w(e_i)}=\frac{n_Z\bullet w(e')\bullet\prod_{i=1}^{|X|}{}_{(\forall e_i\in X)}\, w(e_i)}{N(\sum_{i=1}^{|X|}{}_{(\forall e_i\in X)}\, w(e_i)+w(e'))} \tag{8}$$

If X is infrequent, $wsupport(X)<minSup$, At the same time, $0\le w(e')<1$, $n_Z\le n_X$ so $wsupport(Z)\le wsupport(X)<minSup$, then Z is also infrequent. Therefore, the superset of infrequent sets is infrequent. Similarly, the subset of frequent sets is frequent. The certificate is completed.

A further improvement of this paper is based on the spatial division of the common prefix. Considering the practical significance of the alarm transaction set D, transactions with the same prefix contain similar association properties. That is, the transaction set starting with e_1 reflects the causal association between event e_1 and subsequent events, so we divide the item set according to the prefix, and the item set with the same prefix e_i constitutes the classification subspace, $Q[e_i]$, $Q[e_i]\subset D$. In the process of generating the candidate k-item set C_k according to the frequent k-1 item set L_{k-1} ($k\ge2$). First, the subspace division is performed on the L_{k-1} based on the prefix and then we can obtain a candidate k item set directly by connecting item set in the same subspace. In general, "a subspace can be stored in main memory", so that the work of finding the largest frequent item set is performed on a separate subspace, without comparison and pattern matching, reducing the number of connections, and significantly reducing I/ O overhead.

We name the improved association rule mining algorithm as IWApriori. The algorithm flow is similar to the Apriori algorithm. Firstly, find all frequent item set whose weighted support is not less than the minimum support specified by the user, and then use the frequent item set to generate all the rules that satisfy the minimum confidence. The pseudo code of the algorithm is as follows.

3) Update of association rules----IWApriori-up

The software update and version changes in the cluster system are frequent, which makes the data in the system log dynamically change. In order to obtain the updated association rules, the easiest way is to re-use the Apriori algorithm to mine the database, but not only does the algorithm is less efficient but also does not take full advantage of the results of previous mining. Therefore, this paper proposes the update algorithm IWApriori-up of association rules based on the idea of FUP algorithm. The main idea is to make full use of the original mining rules on the updated database or parameters, find new rules that meet the conditions, and delete the old rules that are invalid that is to minimize the amount of calculation.

We let the original data set be D, and the new data is set be d, then the changed data set is $(D+d)$. At the same time we assume that the IWApriori algorithm has been used to obtain the frequent item set $L(D)$ of the original data set D, and the frequent item set $L(d)$ of the new data set d, then the update of the rules mainly has the following three cases:

(1) If the item set t satisfies $t\in L(d), t\in L(D)$, then $L(D+d)\leftarrow t$.

(2) If the item set t satisfies $t\in L(d), t\notin L(D)$ or $t\in L(D), t\notin L(d)$, the weighted support $support(D+d)$ of t in $(D+d)$ is obtained according to the following equation (9); if $support(D+d)\ge minSup$, then $L(D+d)\leftarrow t$, otherwise t is not a frequent item set.

$$wsupport(t_{D+d})=\frac{wsupport(t_D)\,|D|+wsupport(t_d)\,|d|}{|D|+|d|} \tag{9}$$

(3) If the item set t satisfies $t\notin L(D), t\notin L(d)$, then t must not be a frequent item set.

ALGORITHM IWApriori

INPUT: *D*, *minSup*, *minConf* // *D* is the transaction database, *minsup* is the minimum support, *minConf* the minimum confidence

OUTPUT: *R* // event rules

PROCEDURE:

1: $L=\{\}$, $R=\{\}$, $C=\{\}$

2: scan the DB for $[w(e_i)]$ //scan database *D* for item weight

3: L_1= Generate(*D*); // L_1 is frequent 1-item sets

4: $L=L\cup L_1$

5: **for**(k=2;$L_{k-1}\neq\emptyset; k++$){

6: **if** k==2 **then**

7: C_k=Apriori-gen(L_{k-1}) // Generate frequent 2-item candidate sets

8: **else**

9: Q=get_item_sub_space(L_{k-1})// delimit molecular space

10: **for** each subspace $Q[e_i] \in Q$ {

11: **if** $|Q[e_i]|$=1 **then continue**

12: C_k=Apriori-gen(L_{k-1})} // Generate frequent k-item candidate sets

13: **for** each candidates $c \in C_k$ {

14: $sup(c) = wsupport(c)$ // calculate weighted support

15: **if** $c \neq \emptyset$ &&$sup(c) \geq minSup$ **then**

16: $L = L \cup C$}}

17: **for** each item set $l_l \in L$ {

18: c= caluculate_ $wconfidence\ (l_l)$ // calculate Weighted confidence

19: **if** $c \geq minConfidence$ **then**

20: $R \leftarrow (l_l, s, c)$}

21: Return R

21: **end**

IV. EXPERIMENT

In this section, we will analyze the time performance and spatial performance of the IWApriori association rule mining method proposed in this paper. The comparison method is the rule mining method used by LogMaster [10] and SUCEG . The transaction set used is the Blue Gene/L transaction set. The BlueGene/L data sets were collected from BlueGene/L HPC systems deployed at Lawrence Livermore National Laboratory (LLNL).This data set can be downloaded from CFDR[13].The experimental results are shown in Figs. 2-5.

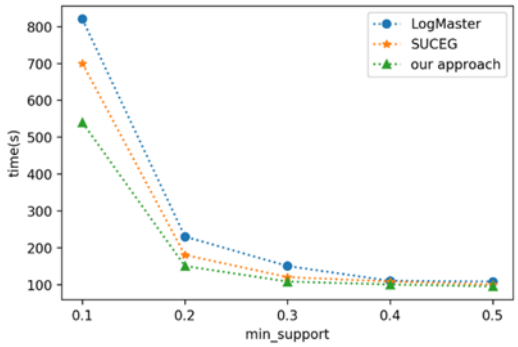

Fig. 2. The execution time of the algorithm with different minimum support

As can be seen from Fig.2, in terms of time performance index, the execution time of each algorithm decreases with the increase of the minimum support threshold. However, under the same support degree, the time performance of the proposed algorithm is better than that of the comparison algorithm, and the smaller the support degree, the greater the advantage. Especially when the minimum weighted support is 0.1, the execution time of IWApriori algorithm in this paper is nearly 30% less than that of LogMaster and 21% less than that of SUCEG. This shows that the improved algorithm has better execution efficiency.

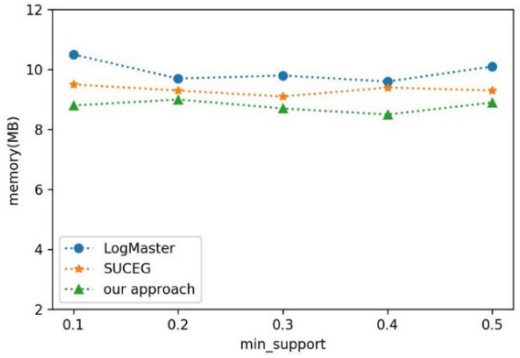

Fig. 3. Memory footprint of algorithms with different minimum support

In terms of spatial performance comparison, Fig.3 shows the memory occupancy of algorithms with different minimum support degrees. The algorithm in this paper adopts the idea of space partition based on common prefix, which increases the sharing of data. Therefore, the spatial performance of this algorithm is better than that of SUCEG and LogMaster algorithms, and the memory occupancy is reduced by 1%-2%.This aspect is very important for real-time system monitoring and prediction, which can reduce the system burden brought by prediction and ensure the normal operation of important services, making fault prediction more realistic.

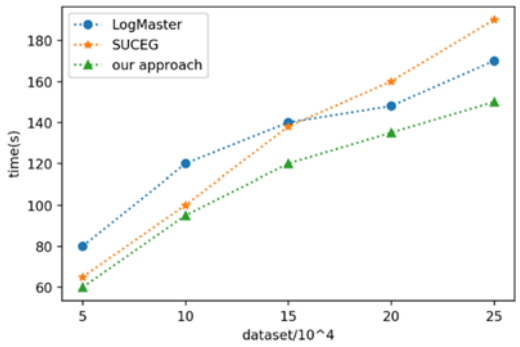

Fig. 4. Time comparison of rule mining during log database update

Fig.4 shows the comparison of mining time of association rules in the process of updating the log database when a new log record is generated. It can be seen that as the database grows (50,000 pieces at a time), the execution time of all methods grows longer, but the method in this paper grows slowly and steadily. This is because IWApriori makes full use of existing mining rules and reduces computation. The mining time of the algorithm in this paper is nearly 20 seconds shorter than that of the LogMaster method. Compared with SUCEG algorithm, the advantages of the method in this paper are not obvious when the database capacity is small, but with the increase of the database capacity, the time performance advantage of the method in this paper in the aspect of rule update is gradually prominent. When there are 250,000 pieces, it can save about 1 minute compared with SUCEG. To sum up, the method in this paper has good mining efficiency and scalability.

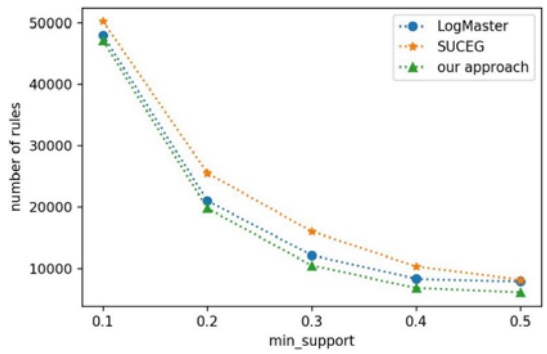

Fig. 5. The number of rules mined by different algorithms

Fig.5 shows the number of rules mined by the algorithm under different minimum support degrees. It can be seen that the number of rules obtained by IWApriori algorithm is about 90% of that obtained by LogMaster and SUCEG, which means that the number of rules extracted by IWApriori algorithm will suffer a certain loss while improving performance, but the loss is within an acceptable range compared with the advantages brought by performance improvement.

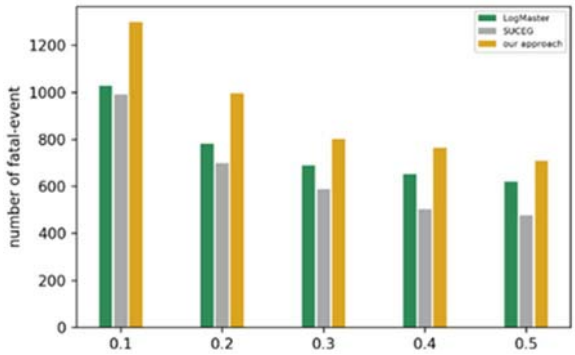

Fig. 6. The number of fatal-event mined by different algorithms

Fig.6 only show the different minimum weighted support algorithm excavated under the circumstance of the number of fatal events, it can be seen that this algorithm is dug up the number of fatal events about one 5 more than SUCEG and LogMaster algorithm, illustrate LogMaster and SUCEG data contains a large number of frequent item sets in high frequency but not important events, it will lead to the operations staff concerned with deadly fault information can not be digging, and this method can eliminate the redundant association rules, mining contains lethal event to more important association rules. To prevent fatal system failure events from being missed and reflected on the indicators is to increase the recall rate.

V. CONCLUSION

The mining of association rules plays an increasingly important role in the intelligent management and system maintenance of large systems. The method of mining and self-updating association rules based on weighted increment is proposed in this paper, which can effectively improve the completeness of rules and realize the efficient mining and updating of rules in the whole life cycle of the system. We used real log data set Blue Gene/L to validate our method. The results show that our association rule mining method is better than other methods in terms of time performance, space performance and the effectiveness of mining rules.

ACKNOWLEDGMENT

This work is supported by National Key Research and Development Program of China(2019YFB2103200),the Industrial Internet Innovation and Development Project 2019 of China("Big data analysis tool software project of Industrial Internet").

REFERENCES

[1] Ghiasvand S, Ciorba F M, Tschüter R, et al.. Lessons learned from spatial and temporal correlation of node failures in high performance computers[C]// Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. IEEE, 2016: 377-381.

[2] Liang Y, Zhang Y, Sivasubramaniam A, et al.. Bluegene/l failure analysis and prediction models[C]//International Conference on Dependable Systems and Networks (DSN'06). IEEE, 2006: 425-434.

[3] Mohammed B, Awan I, Ugail H, et al.. Failure prediction using machine learning in a virtualised HPC system and application[J]. Cluster Computing, 2019, 22(2): 471-485.

[4] Zheng Z, Lan Z, Park B H, et al.. System log pre-processing to improve failure prediction[C]//2009 IEEE/IFIP International conference on Dependable Systems & Networks. IEEE, 2009: 572-577.

[5] Kobayashi S, Otomo K, Fukuda K, et al.. Mining causality of network events in log data[J]. IEEE Transactions on Network and Service Management, 2017, 15(1): 53-67.

[6] Liang Y, Zhang Y, Xiong H, et al.. An adaptive semantic filter for blue gene/l failure log analysis[C]//2007 IEEE International Parallel and Distributed Processing Symposium. IEEE, 2007: 1-8.

[7] Di S, Guo H, Gupta R, et al.. Exploring Properties and Correlations of Fatal Events in a Large-Scale HPC System[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 30(2): 361-374.

[8] Zheng W, Wang Z, Huang H, et al.. A prediction approach for correlated failures in distributed computing systems[C]//2016 IEEE International Conference on Communications (ICC). IEEE, 2016: 1-6.

[9] Gu J, Zheng Z, Lan Z, et al.. Dynamic meta-learning for failure prediction in large-scale systems: A case study[C]//2008 37th International Conference on Parallel Processing. IEEE, 2008: 157-164.

[10] Fu X, Ren R, Zhan J, et al.. LogMaster: mining event correlations in logs of large-scale cluster systems[C]//2012 IEEE 31st Symposium on Reliable Distributed Systems. IEEE, 2012: 71-80.

[11] Fu X, Ren R, McKee S A, et al.. Digging deeper into cluster system logs for failure prediction and root cause diagnosis[C]//2014 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2014: 103-112.

[12] Yu Y, Chen H. An Approach to Failure Prediction in Cluster by Self-updating Cause-and-Effect Graph[C]//International Conference on Cloud Computing. Springer, Cham, 2019: 114-129.

[13] CFDR,Computer Failure Data Repository [Online]. http://www.usenix.org/cfdr-data