Online Event Correlations Analysis in System Logs of Large-Scale Cluster Systems

Wei Zhou, Jianfeng Zhan, Dan Meng, Zhihong Zhang

Online Event Correlations Analysis in System logs of Large-scale Cluster Systems

Wei Zhou[1], Jianfeng Zhan[1], Dan Meng[1], Zhihong Zhang[2]

[1] Institute of Computing Technology, Chinese Academy of Sciences
[2] The Research Institution of China Mobile
zhouwei@ncic.ac.cn

Abstract. It has been long recognized that failure events are correlated, not independent. Previous research efforts have shown the correlation analysis of system logs is helpful to resource allocation, job scheduling and proactive management. However, previous log analysis methods analyze the history logs offline. They fail to capture the dynamic change of system errors and failures. In this paper, we purpose an online log analysis approach to mine event correlations in system logs of large-scale cluster systems. Our contributions are three-fold: first, we analyze the event correlations of system logs of a 260-nodes production Hadoop cluster system, and the result shows that the correlation rules of logs change dramatically in different periods; Second, we present a online log analysis algorithm *Apriori-SO*; third, based on the online event correlations mining, we present an online event prediction method that can predict diversities of failure events with the great detail. The experiment result of a 260-nodes production Hadoop cluster system shows that our online log analysis algorithm can analyze the log streams to obtain event correlation rules in soft real time, and our online event prediction method can achieve higher precision rate and recall rate than the offline log analysis approach.

Keywords: System logs, online log analysis, event correlations, online event prediction

1 Introduction

As the scale of cluster systems grows in the area of scientific computing and commercial applications, *failures* [7] become normal, and their root causes are diversely derived from software, hardware, maintenance (typically by the vendor), operations (management of the system), environment (power, facilities, command lines), and the infrastructure that supports software distribution, and project management [1]. Collected by systems, applications, and tools, *logs* that record important failure events are the first place system administrators go for troubleshooting when they are alerted to a problem. The examples of logs include /dev/error in Linux systems, data collected by tools like OpenView, IBM Tivoli, Microsoft Operations Manager and NetLogger [2].

It has been long recognized that failure events are correlated, not independent. For example, the work published in the year of 1992 [19] has concluded the impact of

correlated failures on dependability is significant. Previous research efforts [3] [4] [5] [6] have shown the correlation analysis of system logs is promising in event prediction and fault diagnosis, and thus it is helpful to resource allocation, job scheduling and system management [3] [5] [9]. Recent work shows the analysis of logs is useful in mining dependency of distributed system [16][17] or model IT service availability [18].

Most of previous log analysis methods are offline. The offline log analysis methods collect log streams of a long period, for example one or even three months, *offline* preprocess logs for mining event correlation or filter events, and then use the analysis result to predict failures or diagnose faults [20]. The offline log analysis methods have three drawbacks: first, it is difficult to provide online service, for example event prediction, to other runtime service such as a job scheduling system. Second, the previous work of Adam Oliner et al [20] shows that over the course of a system's lifetime, anything from software upgrades to minor configuration changes can *dramatically alter the meaning or character of the logs*, while offline log analysis methods are weak in capturing the dynamic of failures. Third, offline tools do not provide the ability to automatically react to problems [21], however, system administrators or autonomic management systems need to deal with the outages in time.

In this paper, we focus on the online log analysis and event prediction, based on the event correlations. *When we refer to an online log analysis method, we indicate three-fold meanings: first, our method can analyze incoming system logs (log stream) in a soft real time; second, almost immediately after an event of cluster system occurs, our method will mine event correlations in almost real time; third, other systems will use mining results for different purposes in time*. In this paper, we will show how to use mining results for event prediction. We also plan to use these results for fault diagnosis or other promising purposes.

The online log analysis of large-scale cluster systems raises several challenges [15]. First, because the meaning or character of the logs changes over the course of the lifetime of a cluster system [20], the analysis algorithm should be suitable for capturing the dynamic nature of logs or failures. Second, the analysis results should be almost real time, accurate and complete for diversities of online systems, for example event prediction, fault diagnoses, job scheduling or checkpoint system etc, and hence other systems can use the online analysis results.

As shown in Fig.1, we treat a continuous time flow as several overlapping time frames; *our system* generates *events rules* that capture the event correlations of logs in different time frames and then updates *the event rule database* that collects event rules mined in the *whole log history*. For example, as shown in Fig.1, at the end of the *Ith time frame*, our system will invoke the job of mining event correlations in the logs of the *Ith time frame,* generate new event rules that capture the event correlations of logs in the *Ith time frame, and then update the event rule database.* In our system, we let the *Ith time frame overlaps the (I+1)th time frame.* This choice is because that *we need to mine the correlations of events in adjacent time frames, and if two adjacent time frames are disjoint, some event correlations will be ignored*.

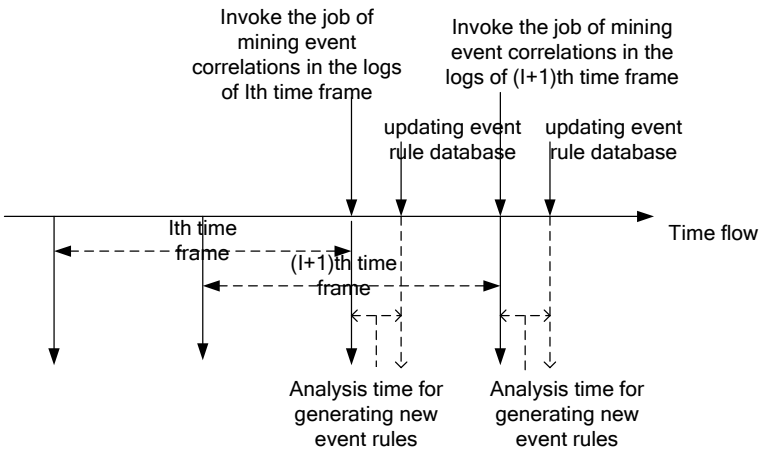

Fig. 1. Basic idea of our online log analysis system.

Our contributions are three-fold. First, we analyze the correlations of logs of a 260-nodes production Hadoop cluster system, and the analysis result shows that the correlation rules of logs change dramatically in different periods; Second, we design and implement the first online log analysis algorithm *Apriori-SO* that supports the online log filtering and event correlations mining; third, based on the online event correlations mining, we present an online event prediction method that can predict *diversities of failure events with the great detail*. We use an emulation methodology to analyze and predict the event logs of a 260-nodes production Hadoop cluster systems in the Research Institution of China Mobile. The experiment result shows that our online log analysis system can analyze the log streams to obtain event correlation rule in soft real time and our online event prediction system can achieve more precision rate and recall rate than our previous offline log analysis system.

The paper is organized as follows: In Section 2, we describe the related work. Section 3 justifies our motivation of the online log analysis. Section 4 presents the design and implementation of our online event correlation mining and online event prediction systems. The evaluation of the system is summarized in Section 5. We draw a conclusion and discuss the future work in Section 6.

2 Related work

Most of traditional log analysis methods are offline. Some work uses statistical analysis approach to find simple temporal and spatial laws or models of system events [6] [5] [10] [11] in large-scale cluster systems like BlueGene/L. When the obtained knowledge is used in event prediction, it may bring high precision rate and recall rate, however it is compared with the filtered logs obtained with the aggressive filtering policy. For example, in the work of [10], 99.96% of original logs are filtered. With

the aggressive filtering policy, the important failure patterns [22] or warning messages [10] , which are often the symptom of fatal errors, may be ignored. Besides, the predicted events are coarse without the detail.

The work of [3] applied time-series algorithms, rule-based classification techniques, and Bayesian network model to assess the effectiveness of these techniques in predicting failure in a cluster. However it either focuses on specific types of failures or targets small scale systems, thus not sufficient for large-scale clusters [6].

Rule-based algorithms are used in some papers [3] [6] [8] [12]. The work of [6] presents a meta-learning method based on statistical analysis and standard association rule algorithm. The rule-based algorithms only consider the correlations between two event types. If we consider the correlations across multiple event types, the precision of event prediction will improve.

Besides, *they did not consider the dynamic change of failure correlations over time*.

3 Motivation: why online log analysis is necessary?

To analyze the log of large-scale cluster systems, in our previous work, we have developed an *offline* log analysis system named *LogMaster* to mine the event correlations of logs.

In this section, we present some offline log analysis results to justify our motivation of online log analysis.

3.1 The description of the Hadoop system

We used *LogMaster* to analyze and predict the logs of a production Hadoop cluster system in the Research Institution of China Mobile, which is the largest telecom operator in the world. The production cluster system is used to run a series of data-intensive applications based on Hadoop [15]. The system has 260 nodes, including 10 *servers* and 250 *data nodes*. *Data nodes* are used to run Hadoop applications, while *servers* are used to analyze logs or manage system.

Inspired by Google's MapReduce and Google File System (GFS), Apache Hadoop is a Java software framework that supports data-intensive distributed applications under a free license. It enables applications to work with thousands of nodes and petabytes of data.

3.2 The introduction of offline log analysis

The detail of the offline log analysis method can be found in our previous work [13]. To save space, in this section, we only give the short description of concepts and methods.

Event preprocessing:

Because of the different sources of event logs, the logs have different formats: text files, databases, or special file formats generated by programs. The log preprocessing step parses the variant log files into a nine-tuples *(timestamp, log id, node id, event id, severity, event type, application name, process id, user)*. The severity degrees include INFO, WARNING, ERROR, FAILURE, FAULT, and the event types include HARDWARE, SYSTEM, APPLICATION, FILESYSTEM, NETWORK, etc. The attributions of *timestamp, node id, application name, process id and user* are easily obtained. The *event id* and *log id* are respectively the mapping functions of 2-tuples (severity, type) and 4-tuple (node id, event id, application, process id). For an upcoming event, the *event id* is generated according to 2-tuples (severity, type). If a new 2-tuples (severity, type) is reported, a new event id will be assigned to this event. If a new 4-tuple (node id, event id, application, process id) is reported, a new *log id* will be assigned to this event.

Event correlations mining:

For an event pair *(A, B), of which event A occurs before event B* within a predefined *sliding time windows of the log buffer* (in short *sliding time window*), we call *event A* is *event B*'s *preceding event*, and *event B* is *event A*'s *posterior event*. *The support count* is the recurring times of the preceding event which is followed by the posterior event, while *the posterior count* is the recurring times of the posterior event which follows the preceding event. For example, for an event sequence *ACBBA*, the support count of (A, B) is one and the posterior count is two. We define the *confidence* of (A, B) as follows:

Confidence (A, B) = support count (A, B) / count (A).

If an event pair occurs within the predefined time window and the support count and confidence of the event pair exceeds the predefined *support count threshold* and *confidence threshold*, we call it an *event rule*.

The time relation of offline log analysis is shown in Fig.2. The Apriori-S algorithm scans the whole log history to get the 2-items event rules using statistical analysis, and generates candidate k-items rules (k>2) based on (k-1)-items rules, then scans the *whole log history* to validate the candidate k-items rules and get k-items rules.

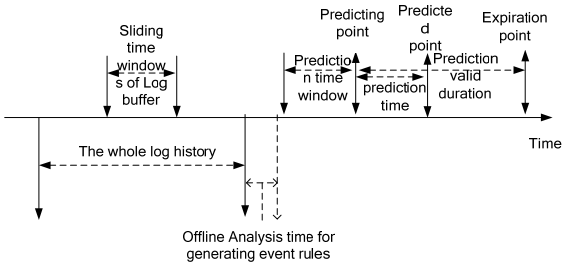

Fig. 2. Time relation in the offline failure analysis approach.

We give an example of 3-item event rules (911->913->985) here. 911, 913 and 985 are three different log id, their details are as follows:

……

Nov 28 13:42:13 compute-5-4.local rpc.statd[2579]: Caught signal 15, un-registering and exiting.

Nov 28 13:44:17 compute-5-4.local sshd[1653]: error: Bind to port 22 on 0.0.0.0 failed: Address already in use.

Nov 28 13:45:33 compute-5-12.local sshd[1655]: error: Bind to port 22 on 0.0.0.0 failed: Address already in use.

Event prediction:

Based on the event rules, a failure predictor can help determine possible occurrences of important events in the near future.

As shown in Fig.2, the system begins predicting failures at the timing of the *predicting point.* The occurring time of predicted failure is called the *predicted point*. The *prediction time* is the time difference between the *predicting point* and the *predicted point*. The *prediction time* is the time span left for the autonomic system or system administrator to respond with possible upcoming failures.

When the system predicts failures at the *predicting point*, the events having occurred at the *prediction time window* will be used to predict events. When using our system to predict failures, the system administrator can predefine the *prediction valid duration* that is the time difference between predicting point and the *expiration point*. If the predicted event occurs within the prediction valid duration, we consider it valid, else we consider it invalid.

Our system can predict failure events with great detail. For example, a predicted event includes the following information: *predicted point, log id, node id, application name, event type, and event severity*.

3.3 Experiment results

Our work analyzes three month's logs of a 260-nodes production Hadoop cluster system in the Research Institution of China Mobile. The logs are collected between Oct 26, 2008 and Dec 31, 2008, which includes 977,858 original event entries.

We use a server to analyze the logs. The server has two Intel Xeon Quad-Core E5405 2.0GHZ processors, 137GB disk, and 8G memory.

We divide three months' logs into three disjoint periods: (1) Period one: from Oct 26, 2008 to Nov 16, 2008; (2) Period two: from Nov 17, 2008 to Dec 08, 2008; (3) Period three: from Dec 09, 2008 to Dec 31, 2008. We independently mine event rules in different periods, and then we compare event rules mined in different periods. The comparison of event rules is shown in Table 1. From Table 1, we can observe that the event rules dramatically change in different periods. For example, there are only 12 same events rules occurring in both *Period one and Period two*. This has several possible causes: (1) the repair of the failure or the self-healing of systems or applications cause some failure events disappear; (2) changes of events in the system or applications; (3) new coming failure events.

Table 1. The comparison of event rules in different periods (log buffer =60 minutes, support count threshold =5, confidence threshold =0.5).

number of same event rules	period 1	period 2	period 3
period 1	133	12	5
period 2		51	6
period 3			48

Finally, we use the event rules obtained from *Period I* to predict events in *Period J*. The experiment result is shown in Fig.3. From Fig.3, we can see that if we use event rules in a period to predict events in another period, the precision rate is low. However, if we use event rules in a period to predict events *in the same period, the precision rate is higher*.

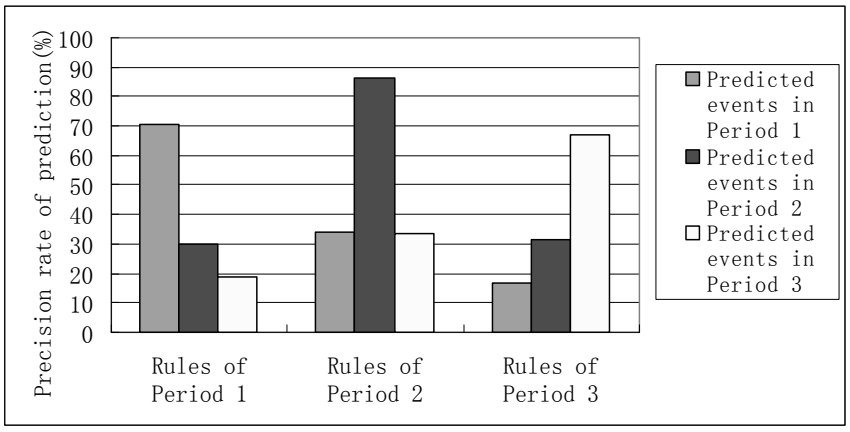

Fig. 3. The precision rate of event prediction using vent rules in different period. (Log buffer =60 minutes, prediction time window = 60 minutes, support count threshold =5, confidence threshold =0.5). The precision rate of event prediction is defined in Section 5.1.

Our observation is complementary to the previous work of Adam Oliner et al [20]. The previous work of Adam Oliner et al show that over the course of a system's lifetime, anything from software upgrades to minor configuration changes can *dramatically alter the meaning or character of the logs*. Both observations justify our motivation of online log analysis.

4 Online log analysis and event prediction

In this section, we introduce the online log analysis algorithm and the online event prediction method. In this section, *we use the same concepts introduced in Section 3.2*.

4.1 Online log analysis

Different from Apriori-S algorithm described in our previous work [13], we use a sliding time window model as shown in Fig.4 so as to analyze the event log streams. The logs in the sliding time window are saved into the log buffer. When all the logs in the time windows are read into the log buffer, the online log analysis system will analyze the logs in the two adjacent log buffers to generate new event rules and then update the event rule base.

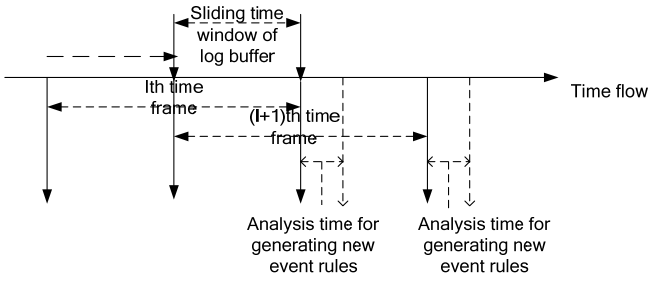

Fig. 4. Time relation in our online failure analysis approach.

4.2 Apriori-SO event correlations mining algorithm

For event logs in multiple nodes of cluster systems, we take the following facts into accounts: (1) the replicated applications in multiple nodes may have the same errors or software bugs, and failure events may appear in multiple nodes; (2) nodes in large-scale computing systems transfer data and communicate with each other, so a failure on one node may cause related *failures* on other nodes; (3) a failure on one node may change the environment of the system, which may cause other *failures* on other nodes. So as to analyze the correlation of failure events among multiple nodes, we use a log filtering policy that only analyzes events occurring in the same node or having the same event types or of the same applications. It can effectively reduce the size of the analyzed logs and decrease the *analysis time*.

On the basis of the Apriori associate rule mining algorithms [13], we propose an improved algorithm, named *Apriori-SO*, to get the frequent itemsets with the support count *above* the user-defined threshold value.

The Apriori-SO algorithm is a one-pass algorithm, which is described as below:

(1) Proper thresholds of *support Sth* and *confidence Cth* are predefined; the proper sliding time window *Tb* is defined too.

(2) Suppose *B* is the current log buffer, and *Bp* is the preceding log buffer. Suppose *C (k)* means the set of frequent k-items event set candidates, *F (k)* means the set of frequent k-item event set, *R (k)* means the set of k-item event rules. Set *R (2)* = {};

(3) If a new event *e* comes, add *e* into the log buffer *B*. If all events in the current time window is read into the current log buffer, goto step (4); else loop step (3).

(4) Scan the log buffer B and Bp, count the number of each event, and *support count* and *posterior count* of each event pair *(i , j)* while *i* and *j* are both in *B*,or *i* in *Bp* and *j* in *B*.

(5) Calculate cumulative number of each event, and support and confidence of each event pair *(i, j)*. Update the support and posterior of event rules in *R (2)*, and add new event rules if the support and confidence of an event pair are above thresholds.

(6) Get new frequent k-items *(k>=3)* event set candidates. If two adjacent subsets of a k-items event set are in *F (k-1)*, add the k-items event set into *C (k)*. For example, if *(A,B)* and *(B,C)* are frequent 2-items event set, then the 3-items event set *(A,B,C)* are frequent 3-items event set candidates and add it into C(3).

(7) Scan the *B* and *Bp* to get the support count and posterior count of event pairs in *C (k)*. We regard the log buffer *B* and *Bp* as the sampling of the log history, and calculate the approximate value of support count and posterior count of each event pair in *C (k)*;

(8) Add the k-items candidates in *C (k)* having support count above the threshold to *F (k)*, and add the k-items in *F (k)* having confidence above the threshold to *R (k)*;

The 2-items event rules are generated in step (2) and step (3). In the step (2) and step (3), we only consider the event set that have the same *node name* or *event type* or *of the same application name* occurring in the time window *Tb* which is defined in step (1). This event filtering policy can reduce the amount of event logs effectively.

Based on the 2-items rules, the k-items rules are generated in step (4). Different from the Apriori-S algorithm which get the support of the candidate rules by scanning the *whole log history* (Shown in Fig. 2), we get the approximation value of support count of k-items rules based on the support count of (k-1)-items rules.

4.3 Online event prediction

The online event prediction is shown in Fig.5. The concepts of *predicting point*, *predicted point*, prediction valid duration, *prediction time*, and *prediction time window are same like that of offline log analysis, which are explained in Section 3.2.*

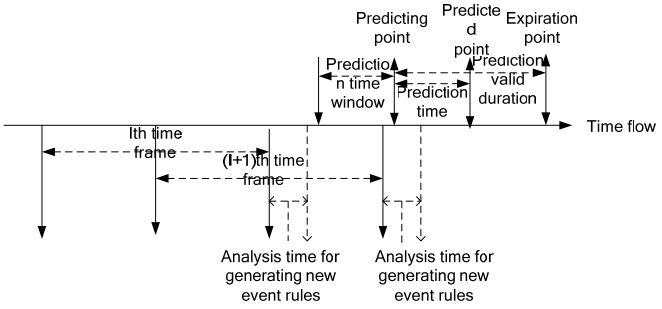

Fig. 5. Time relation of our online event prediction approach.

As shown in Fig.5, the difference of the online event prediction method from our previous offline one is that at the end of each time frame, we will generate new event rules and update the event rule database. Finally, we use the updated event rules to predict upcoming events. The implementation of our online event prediction system is same like that in our previous work [13].

We find all the correlated events of coming log in the prediction time window. The current state of system is determined according to these events, and the upcoming correlated events are predicted based on the event rules.

5 Experiments

The target Hadoop system and its system logs in our experiment are described in Section 3.2. In our experiment, we preprocess and analyze the logs between Oct 26, 2008 and Dec 31, 2008. The event rules generated according to the event logs are used to online predict the event logs in Jan, 2009.

5.1 Metrics

We use *the analysis time*, *the memory usage of the Log server node* to evaluate the overhead of our online log analysis system, and compare *the precision rate*, *the recall rate*, and *the average prediction time* to evaluate our online event prediction system.

(1) Average analysis time

The compute complexity means the time and space utilization of algorithm, including time complexity and space complexity. In this experiment, we use the *average analysis time and the average analysis time of time frames* to evaluate the compute complexity.

As shown in Fig.5, *the analysis time* is the time difference between the beginning and ending timing points of event preprocessing, filtering and correlations mining.

Average analysis time = the total analysis time / count of event logs

Average analysis time of time frames = the total analysis time / count of time frames

(2) Precision rate and Recall rate

The *precision rate* means the ratio of the correctly predicted events to all predicted events. The *recall rate* means the ratio of correctly predicted events to all forthcoming events.

True Positive (TP) = the count of events which are correctly predicted

False Positive (FP) = the count of events which are predicted but not appeared in the time window

Precision rate = *TP* / (*TP* + *FP*)

Recall rate = *True Positive*/ *count of all events*

(3) Average prediction time

The prediction time is defined in Section 3.2. The *prediction time* is the time span left for the autonomic system or system administrator to respond with the possible upcoming failures.

5.2 The experimental methodology

Our online event correlation system includes two major components: *Log agents* and *Log server*. *Log agents* in each data node collect logs and transfer logs to *Log server* in almost real time. After *Log server* receives all the logs of a time frame, it will preprocess and filter the log stream, and mine the new event rules.

In our experiment, we use *an emulation methodology*. Instead of log agents on each data node sending logs to the log server, we use a program on another node to replay the logs of the 260-nodes Hadoop cluster systems, and send the logs to *Log server* in real time according to the timestamp of each event in logs.

5.3 Online log analysis

After log preprocessing and filtering, the event logs are used to analyze the failure correlations.

When the 38432 logs between Oct 26, 2008 and Dec 31, 2008 are analyzed by our *Apriori-SO* algorithm, the support threshold and the confidence threshold are respectively set to 0.5 and 0.25 through comparing several runs of experiments with different configurations.

The *average analysis time* and the analysis time of time frames are shown in Fig.6 with the varying time windows of the log buffer. We can see from Fig.6 that the average analysis time of time frames increases with the sliding time window of the log buffer, and the average analysis time decreases.

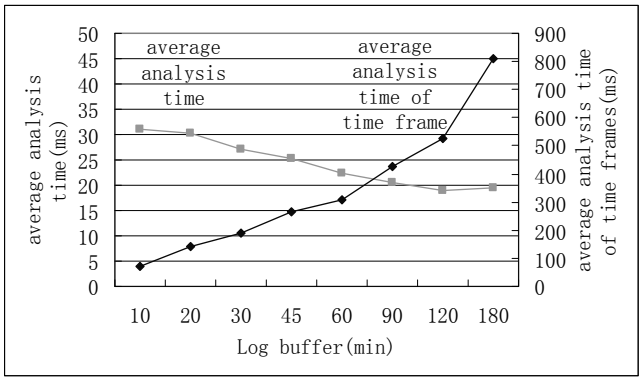

Fig. 6. Average analysis time and average analysis time of time frames V.S. time window of Log buffer Tb (Sth=5, Cth=0.25).

The number of event rules is shown in Fig.7 with the varying time window of the log buffer. We can observe in Fig.7 that the number of event rules also increases with the size of log buffer. That is to say, the integrity of associate event rules is related to the increase of the set of log buffer.

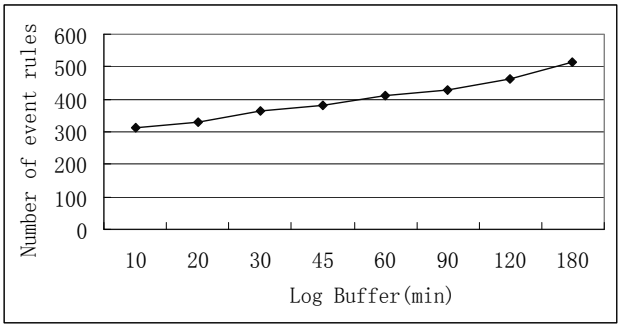

Fig. 7. Number of event rules V.S. Log buffer *Tb* (Sth=5, Cth=0.25).

5.4 Online event prediction

Together with the online log analysis, online event prediction is used to predict upcoming events based on the event rules. Based on event rules generated with the Apriori-SO algorithm, the logs are used to online predict events. In our experiment, we online analyze the event logs between Oct, 2008 and Dec, 2008, and then use these event rules to online predict the event logs in Jan, 2009.

When the support threshold and confidence threshold are respectively set to 5 and 0.25, and the prediction valid duration of online prediction is set as 60 minutes, the result is shown in Fig.8.

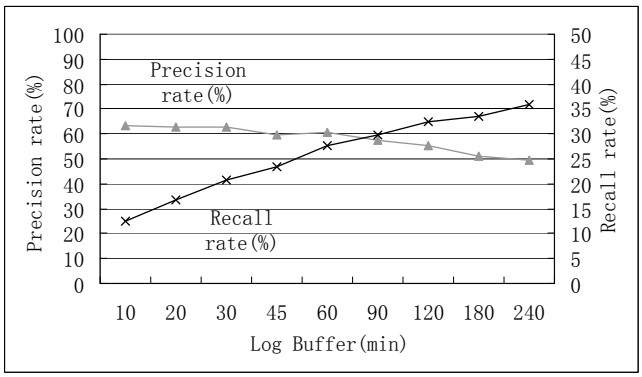

Fig. 8. Precision rate and Recall rate of online prediction V.S. Log Buffer (Sth=5, Cth=0.25, Prediction valid duration=60 minutes)

When the log buffer of online log analysis is set as 60 minutes, the support threshold and confidence threshold are respectively set to 5 and 0.25. The result shows that the number of event rules increases from 355 to 386.

In Fig.9 and Fig.10, we compare our online prediction approach with our previous prediction approach based on the offline log analysis. Fig.9 presents the relationship between the precision rate and the prediction valid duration for both online and offline event prediction. Fig.10 presents the relationship between the recall rate and the prediction valid duration for both online and offline event prediction.

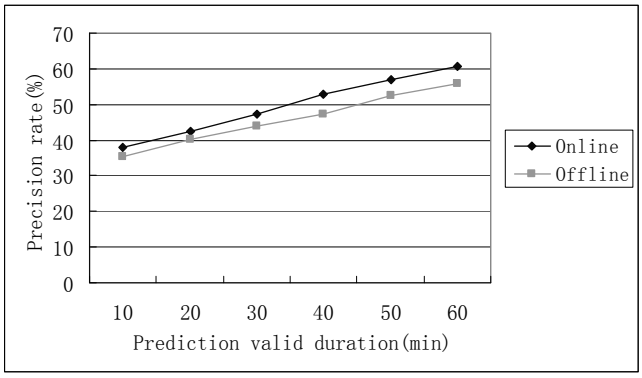

Fig. 9. Precision rate of online and offline prediction V.S. Prediction valid duration (log buffer=60 minutes, Sth=5, Cth=0.25)

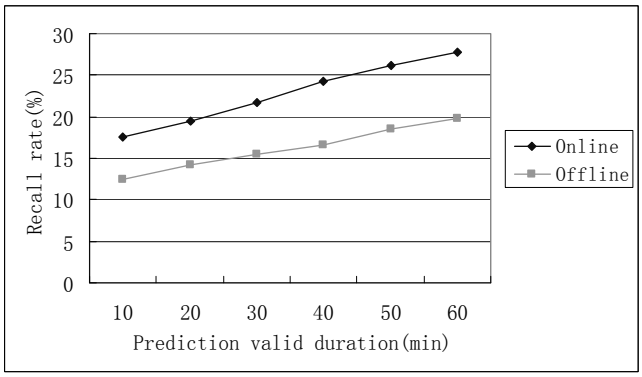

Fig. 10. Recall rate of online and offline prediction V.S. Prediction valid duration (log buffer=60 minutes, Sth=5, Cth=0.25)

It can be seen from Fig.9 and Fig.10 that the precision rate and recall rate of online prediction are higher than offline prediction. It demonstrates that the online log analysis can effectively represent the dynamic change of event rules.

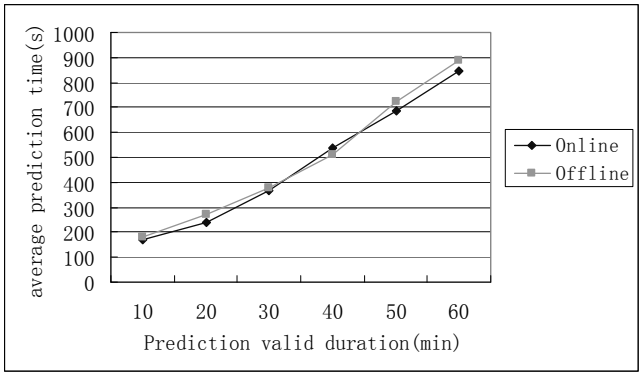

Fig. 11. Average prediction time of online and offline prediction V.S. Prediction valid duration (log buffer=60 minutes, Sth=5, Cth=0.25)

The relationship between the average prediction time and prediction valid duration is shown in Fig.11. The difference between the average prediction time of online prediction and that of offline prediction are small.

6 Conclusion

In this paper, we present an online log analysis approach to analyze event logs of large-scale cluster systems. After online preprocessing and filtering, filtered logs are used to mine failure correlations and generate associate event rules online. We propose an improved *Apriori* associate rules algorithm named *Apriori-SO* to analyze the event logs. The *Apriori-SO* algorithm uses the sliding time window model and stores the log streams to log buffer, and analyze the log buffer to generate new rules and update the existed rules. Based on the observation that most of events rules occur in the same nodes or applications or have the same types, the Apriori-SO algorithm use an event filtering policy to reduce the computing complexity.

The event rules generated by Apriori-SO algorithm can be used in online event prediction. The experiments on a production cluster system in the Research Institution of China Mobile show that our algorithms can achieve good precision rate in event prediction.

In the near future, we will integrate path-based request tracing [22] and event correlation mining approaches to diagnosis the failure events and performance problems of Internet services applications. Much work will be done to analyze multi-dimensional event logs in large-scale cluster systems [23].

Acknowledgments. This paper is supported by the NSFC projects (Grant No. 60703020 and Grant No. 60933003).

References

1. Sahoo, R.K., Sivasubramaniam, A., Squillante, M.S.: Failure data analysis of a large-scale heterogeneous server environment. In: *Proc. of DSN*, 2004
2. Tierney, B., Johnston, W.: The NetLogger methodology for high performance distributed systems performance analysis. In: *Proc. of HPDC*, 1998
3. Sahoo, R.K., Oliner, A.J.: Critical Event Prediction for Proactive Management in Large scale Computer Clusters. In: *Proc. of SIGKDD*, 2003
4. Fu, S., Xu, C.: Exploring Event Correlation for Event prediction in Coalitions of Clusters. In: *Proc. of ICS*, 2007
5. Fu, S., Xu, C.: Quantifying Temporal and Spatial Correlation of Failure Events for Proactive Management. In: *Proc. of SRDS*, 2007
6. Gujrati, P., Li, Y., Lan, Z.: A Meta-Learning Failure Predictor for Blue Gene/L Systems. In: *Proc. of ICPP*, 2007
7. Knight, J.C.: An Introduction To Computing System Dependability. In: *Proc. of ICSE*, 2004
8. Tang, D., Iyer, R.K.: Analysis and Modeling of Correlated Failures in Multicomputer Systems. In: *IEEE Trans. on Comput*. 41, 5 (May. 1992), 567-577.
9. Koskinen, E., Jannotti, J.: BorderPatrol: Isolating Events for Precise Black-box Tracing. In: *Proc of Eurosys*, 2008
10. Liang, Y., Zhang, Y.: BlueGene/L Failure Analysis and Prediction Models. In: *Proc. of DSN*, 2006
11. Hacker, T.J., Romero, F., Carothers, C.D.: An analysis of clustered failures on large supercomputing systems. In: *Journal of Parallel and Distributed Computing*. 69, 7 (Jul. 2009), 652-665.
12. Oliner, A.J., Aiken, A., Stearley, J.: Alert Detection in Logs. In: *Proc. of ICDM*, 2008
13. Zhou, W., Zhan, J., Meng, D., Xu, D., Zhang, Z.:LogMaster: Mining Event Correlations in Logs of Large-scale Cluster Systems. In: CoRR abs/1003.0951: (2010)
14. Jiang, N., Gruenwald, L.: Research Issues in Data Stream Association Rule Mining, In: *ACM SIGMOD Record*, Vol. 35, No. 1, Mar. 2006
15. http://en.wikipedia.org/wiki/Hadoop
16. Salfner, F., Tschirpke, S.: Error Log Processing for Accurate Event prediction. In: *USENIX workshop on the analysis of System logs (WASL)*, 2008
17. Lou, J.G., FU, Q., Wang, Y., Li, J.: Mining Dependency in Distributed Systems through Unstructured Logs Analysis, In: *USENIX workshop on WASL*, 2009
18. Zhang, R., Cope, E., Heusler, L., Cheng. F.: A Bayesian Network Approach to Modeling IT Service Availability using System Logs. In: *USENIX workshop on WASL*, 2009
19. Tang, D., Iyer, R.K.: Analysis and Modeling of Correlated Failures in Multicomputer Systems. In: *IEEE Trans. on Comput.* 41, 5 (May. 1992), 567-577.
20. Oliner, A., Stearley, J.: What Supercomputers Say: A Study of Five System Logs. In: *Proc of DSN*, 2005
21. Rouillard, J.P.: Real-time log file analysis using the Simple Event Correlator (SEC). In: *Proc of LISA*, 2004
22. Zhang, Z., Zhan, J.:Precise request tracing and performance debugging of multi-tier services of black boxes. In: *Proc of DSN*, 2009.
23. Zhou, W., Zhan, J.: Multidimensional Analysis of System Logs in Large-scale Cluster Systems. In: *Proc of DSN (FastAbstract)*, 2008.