Expert Systems with Applications 42 (2015) 7012–7025

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Node anomaly detection for homogeneous distributed environments

Jian Xu [a], Yexi Jiang [b], Chunqiu Zeng [b], Tao Li [b,c,*]

[a] School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing, China
[b] School of Computing and Information Sciences, Florida International University, Miami, FL, USA
[c] School of Computer Science & Technology, Nanjing University of Posts and Telecommunications (NJUPT), Nanjing, China

ARTICLE INFO

Article history:
Available online 5 May 2015

Keywords:
Anomaly detection
Multiple data streams
Contextual anomaly
Collective anomaly

ABSTRACT

Identifying the anomalies is a critical task to maintain the uptime of the monitored distributed systems. For this reason, the trace data collected from real time monitors are often provided in form of streams for anomaly detection. Due to the dramatic increase of the scale of modern distributed systems, it is challenging to effectively and efficiently discover the anomalies from a voluminous amount of noisy and high-dimensional data streams. Moreover, the evolving of the system infrastructures brings new anomaly types that cannot be generalized as existing ones, making the existing anomaly detection solutions unavailable.

To address these issues, in this paper, we introduce a new type of anomalies called contextual collective anomaly. Then we propose a framework to discover this type of anomaly over a collection of data streams in real time. A primary advantage of this solution is that it can accurately identify the anomalies by taking both the contextual information and the historical information of a data stream into consideration. Also, the proposed framework is designed in a way with a low computational cost, and is able to handle large-scale data streams. To demonstrate the effectiveness and efficiency of our proposed framework, we empirically validate it on a real world cluster.

1. Introduction

A homogeneous distributed environment generally consists of multiple computing nodes with the same hardware configuration, software environment and similar workloads. A typical example of the homogeneous distributed environment is the load-balanced system, which is widely used at the backend by the popular large-scale web sites like Amazon, Google and Facebook. In such a distributed environment, the computing nodes in a distributed system would behave similar to each other in the ideal situation (no anomaly and no occasional fluctuation). In such a situation, the observations (in terms of monitored metrics) of the nodes should be close to each other at any time. In practice, node anomaly might be caused by a variety of reasons, such as software aging, resource contention, and hardware failure, making the affected nodes behave differently from other nodes (Grottke & Trivedi, 2007). Overtime, a system is becoming more instable and it would fail to function properly due to the existence of anomaly nodes. Although the health-related data are collected across the system for troubleshooting, unfortunately how to effectively and efficiently identify anomalies and their root causes in the data has never been as straightforward as one would expect.

Traditionally, domain experts are responsible for examining the data with their experience and expertise. Such a manual process is time-consuming, error-prone, and even worse, not scalable. Due to the data scale and complexity, event the domain experts cannot fully identify the true anomalies and may also miss some deeply hidden anomalies. Moreover, as the behaviors of the distributed environment are likely changing over time, such temporal dynamics is difficult to be captured by the domain experts, as they may not be able to refresh their knowledge quick enough.

As the size and complexity of computer systems continue to grow, the difficulty for automated anomaly identification increases dramatically and it have far beyond the processing capability of the domain experts. The traditional expert systems that encoded the rules of the domain experts can only partially addressed the data scale problem. However, they cannot solve the complexity problem. This is because a distributed environment is dynamic. It is not likely such changing behaviors can be well captured by the static rules. Overtime, the deployed expert system based anomaly detection would gradually be outdated, as the rate of false positive and false negative would increase.

There are quite a few data processing and anomaly analysis infrastructures to enable automated anomaly identification.

* Corresponding author.
E-mail addresses: dolphin.xu@njust.edu.cn (J. Xu), yjian004@cs.fiu.edu (Y. Jiang), czeng001@cs.fiu.edu (C. Zeng), taoli@cs.fiu.edu (T. Li).

http://dx.doi.org/10.1016/j.eswa.2015.04.037
0957-4174/© 2015 Elsevier Ltd. All rights reserved.

However, these existing data processing infrastructures are designed based on inherent non-stream programming paradigm such as Map/Reduce (Dean & Ghemawat, 2008), Bulk Synchronous Parallel (BSP) (Valiant, 1990), and their variations. To reduce the processing delay, these applications have gradually migrated to stream processing engines (Arasu et al., 2003;Chandrasekaran et al., 2003). As the infrastructures have been changed, anomalies in these applications are required to be identified online across multiple data streams. The new data characteristics and analysis requirements make existing anomaly detection solutions no longer suitable.

To address the problem, in this paper, we present a real time mechanism for node anomaly detection by taking both the node context information and the node historical information into consideration from multiple data streams.

1.1. A motivating example

Fig. 1 illustrates the scenario of monitoring a 6-node computer cluster, where the x-axis denotes the time and the y-axis denotes the CPU utilization. The cluster has been monitored during time $[0, t_6]$. At time t_2, a computing task has been submitted to the cluster and the cluster finishes this task at time t_4. As shown in Fig. 1, two nodes (marked in dashed line) behave differently from the majority during some specific time periods. Node 1 has a high CPU utilization during $[t_1, t_2]$ and a low CPU utilization during $[t_3, t_4]$ while node 2 has a medium CPU utilization all the time. These two nodes with their associated abnormal periods are regarded as anomalies. Besides these two obvious anomalies, there is a slight delay on node 3 due to the network delay and a transient fluctuation on node 4 due to some random factors. However, they are normal phenomena in distributed systems and are not regarded as anomalies.

A quick solution for stream based anomaly detection is to leverage the techniques of complex event processing (CEP) [3,4] by expressing the anomalies detection rules with corresponding continuous query statements. This rule-based detection method can be applied to the scenarios where the anomaly can be clearly defined. Besides using CEP, several stream based anomaly detection algorithms have also been proposed. They either focus on identifying the contextual anomaly over a collection of stable streams (Bu, Chen, Fu, & Liu, 2009) or the collective anomaly from one stream (Anguilli & Fassetti, 2007; Pokrajac, Lazarevic, & Latecki, 2007). These existing methods are useful in many applications but they still cannot identify certain types of anomalies.

Fig. 2 plots the ground truth as well as all the anomalies identified by existing methods including the CEP query with three different rules (Rule-CQ1, 2, and 3), the collective based anomaly detection (Breunig, Kriegel, Ng, & Sander, 2000), and contextual based anomaly detection (Chandola, Banerjee, & Kumar, 2009).

To detect the anomalies via CEP query, the idea is to capture the events when the CPU utilizations of nodes are too high or too low. An example query following the syntax of Agrawal, Diao, Gyllstrom, and Immerman (2008) can be written as follows:

PATTERN SEQ(Observation o[])
WHERE avg(o[].cpu) oper threshold (AND|OR avg(o[].cpu) oper threshold)*
WITHIN {***length of sliding window***}

where the selection condition in ***WHERE*** clause is the conjunction of one or more boolean expressions, ***oper*** is one of f $>, <, <>$, ==g, and **threshold** can be replaced by any valid expression. However, CEP queries are unable to correctly identify the anomalies in Fig. 1 no matter how the selection conditions are specified. For instance, setting the condition as ***avg*(o[].cpu) >** {**threshold**} would miss the anomalies during $[t_3, t_4]$ (Rule-CQ1); setting the condition as ***avg*(o[].cpu) <** {**threshold**} would miss the anomalies during $[t_1, t_2]$ (Rule-CQ2); and combining the two above expressions with OR still does not work (Rule-CQ3). Besides deciding the selection condition, how to rule out the situations of slight delays and transient fluctuations, and how to set the length of the sliding windows are all difficult problems when writing the continuous queries. The main reason is that the continuous query statement is not suitable to capture the contextual information where the "normal" behaviors are also dynamic (the utilizations of normal nodes also change over time in Fig. 1).

Compared with CEP based methods, contextual anomaly detection methods (such as Gupta, Sharma, Chen, & Jiang, 2013; Jiang, Chen, & Yoshihira, 2006) achieve a better accuracy as they utilize the contextual information of all the streams. However, one limitation of contextual based methods is that they do not leverage the temporal information of streams and are not suitable for anomaly detection in dynamic environments.

Therefore, these methods would wrongly identify the slightly delayed and fluctuated nodes as anomalies. For the given example, collective anomaly detection methods do not work well neither. This is because these methods would identify the anomaly of each stream based on its normal behaviors. Once the current behavior of a stream is different from its normal behaviors (identified based on historical data), it is considered as abnormal. In the example, when the cluster works on the task during time period $[t_3, t_4]$, all the working nodes would be identified as abnormal due to the sudden burst.

1.2. Contributions

In this paper, we propose an efficient solution to identify this special type of anomaly in the above example, named contextual collective anomaly. Contextual collective anomalies bear the characteristics of both contextual anomalies and collective anomalies.

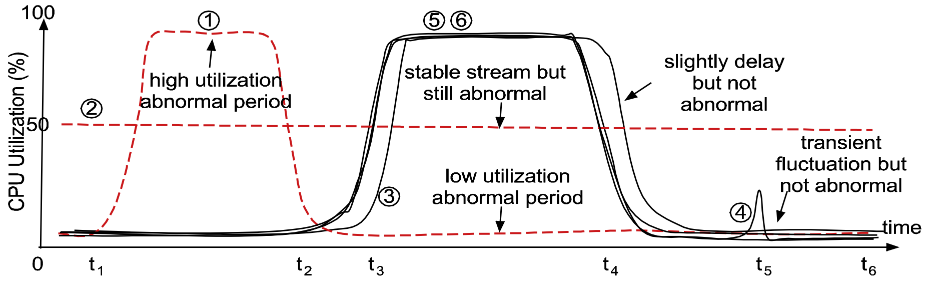

Fig. 1. CPU utilization of a computing cluster.