International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

A Survey on Automatic Fault Detection Framework for Cloud based Application

Kshitija Nandgaonkar[1], Swarupa Kamble[2]

[1]Dept. of Computer Engineering, RMD Sinhgad School of Engineering, Pune, Maharashtra, India

[2]Assistant Professor, Dept. of Computer Engineering, RMD Sinhgad School of Engineering, Pune, Maharashtra, India

Abstract: Cloud computing becoming an emerging and important platform for web service applications. In modern era Cloud Computing has its own challenges. Sometimes it's the fluctuating workload which presses the application to different and unexpected condition. Also the behavior of the cloud hosted application makes it difficult to understand the root cause of failures which results in loss of time and money. Even if the risks are known , it becomes very difficult for the manual operators to resolve them, as an application could go down if things are not done properly. For addressing these issues, this paper proposes an automatic failure detection framework for the cloud based applications. In particular, we introduce an clustering method that captures workload fluctuation and model the correlation between the workloads and application performance.

Keywords: Cloud computing, Web application, fault diagnosis, workload fluctuation, canonical correlation analysis**.**

1. Introduction

Cloud Computing is one of the most used technology in ecommerce sector. In recent years, many web applications are deployed on public cloud computing platforms. But the development and deployment of Web applications are vulnerable to many types due to the complexity, dynamism and openness of cloud computing. These faults cause an application failure which affects a large population of users and results in heavy economic loss. Detecting faults accounts for 75% of failure recovery time, and detecting faults in time could prevent 65% of failures, according to the report of Tellme Networks [1]. Thus, fault diagnosis is essential to guarantee the reliability and performance of Internet-based services. However, the faults inside Web applications (e.g., resource contention, configuration faults, software faults, and hardware failures) are usually triggered by complex factors in the deployment environment at runtime. This makes it difficult to reproduce the faults (e.g., deadlock caused by concurrency and fault transmission between components).
Therefore, debugging and testing Web applications cannot effectively eliminate the inevitable faults triggered in specific contexts.

Fault diagnosis technologies have widely attracted the attention of industrial and academic communities in recent years. For example, commercial monitoring tools (e.g., IBM Tivoli, HP OpenView, and Amazon CloudWatch) allow system operators to set rules for some particular system metrics manually. Then they raise alerts automatically when the value of a metric exceeds the predefined threshold. However, setting suitable thresholds for thousands of metrics in various Web applications is difficult. Similarly, by modeling the status of distributed systems during their normal operation periods, traditional fault diagnosis methods can detect faults when the monitored status deviates from the built model [2]. However, existing solutions are difficult to detect faults in Web applications deployed in a large-scale dynamic cloud computing environment due to the following reasons.

First, cloud computing systems usually provide services using a virtualization technique which causes an application models to be changed from time to time with dynamic workloads and resource allocation. This will be very challenging for fault diagnosis methods to model the status of applications and differentiate their abnormal behaviors from normal ones.

Second, Web applications are often set up in a large scale data center consist of thousands of nodes. Thus fault diagnosis methods found it difficult to model applications involving a large number of metrics taken from many layers.

For addressing such issues, we propose a Fault Diagnosis Framework for Web Applications in cloud Environment, which automatically detect faults and locate their root causes.

2. Overview

In particular, we introduce an online incremental clustering method that will capture workload fluctuation, and make a use of canonical correlation analysis (CCA) which model the correlation between the workloads and the metrics of application performance/resource utilization in each particular access behavior pattern.

The main contributions of this paper are as follows:

- We propose an online incremental clustering method to comprehend the access pattern of customers. As a result, the accuracy of fault diagnosis in each specific access behavior pattern can be improved significantly.
- We use CCA to model the correlations between workloads and metrics automatically.
- We introduce a fault diagnosis conceptual framework which can be mostly used to diagnose faults inside Web applications deployed in cloud computing environment.

3. Existing Methodology

Fault detection/diagnosis for distributed systems has acquired a large attention in academic and industrial areas. Existing methods involves a signature-based methods and anomaly detection methods (including behavior-based methods and metric correlation-based methods). Nowadays, some comparable studies are conducted on cloud computing systems. Additionally, some scientists also focus only on performance metrics, and introduce performance anomaly detection methods. In this section, we methodically study these methods, and discuss their pros and cons.

3.1 Signature-Based Fault Diagnosis

In this category, detection of fault depends upon a specific set of rules. In short it define the signatures of known faults, and detect faults by matching a specific set of rules.

For example, system operators usually collects monitoring data, set of rules and generates an alert using commercial monitoring tools like HP OpenView and IBM Tioli. Whenever values of metrics exceed their predefined threshold values, alerts are generated automatically. However, in complex system it is very difficult task of setting appropriate thresholds for so many metrics. Chen *et al.* [5] stored historical failures and retrieved similar instances in the occurrence of a failure. The failure characteristics were described as an invariant network [5]. These methods are effectual when the signatures of faults are properly defined.

But, it is difficult to identify unknown fault separately, however, our method can detect unknown faults by using correlation model between workloads and metrics based on monitoring data. This can be achieved by tracking the correlation change without prior knowledge about applications and symptoms.

3.2 Behavior-Based Fault Diagnosis

In this category the behaviors of applications is modeled (e.g., component interactions and execution paths), and detect faults by understanding the behavior deviation from the built model. For example, Chen *et al.* [6] represented the execution paths using a probabilistic context-free grammar.

The paths were considered as anomalies if they cannot be parsed by the grammars [6]. Barham *et al.* [7] used clustering to group paths, and the ones were treated as abnormal if they did not fit the built clusters. Chen *et al.* [8] used statistics to periodically analyze interactions between one component and the others using $\chi 2$-test. These methods have capacity to detect application-level faults. However, they cannot detect the faults caused by resource contention. In this paper, we analyze the system-level metrics which indicate the resource utilization. Therefore, our method is capable of locating the suspicious resource utilization which probably causes the faults.

3.3 Metric Correlation-Based Fault Diagnosis

This type of methods generates alerts when the represented hidden invariant relationships among system metrics were lost. For example, Jiang *et al.* [9] collected the metric correlations with the autoregressive linear regression with exogenous input models, and proposed two algorithms that speed up the discovery of correlations. However, the above methods can be applied to diagnose faults inside some applications without domain knowledge, but modeling various correlations among a large amount of metrics in complex systems is difficult. As the workload fluctuates and the access behavior pattern varies, the metric correlation changes significantly.

3.4 Performance Anomaly Detection

Methods of this category pay attention to system performance [3]–[4]. They built models to predict the expected performance metrics, and compared them with the online monitored ones. Such methods require domain knowledge (e.g., the system internal structure) and accurate parameters (e.g., component service time), which are difficult to obtain in practice. Similarly, we can use a tree augmented naïve Bayes (TAN) to identify which system-level metrics were correlated with the high-level performance service level object (SLO) violations. The work aims at finding critical metrics which have an important impact on SLO instead of tracking the system status to detect anomalies. Differently, our method automatically models correlations and detects unseen anomalies by analyzing historical data without domain knowledge and fine-grained parameter estimation.

4. Conceptual Framework

This section describes the conceptual framework of our automatic fault diagnosis method, which can be widely used in cloud computing systems to diagnose faults inside Web applications. There are four components in a fault diagnosis method, which includes system monitoring, status characterization, fault detection, and fault localization. Accordingly, this framework is composed of five components, as shown in Fig. 1.

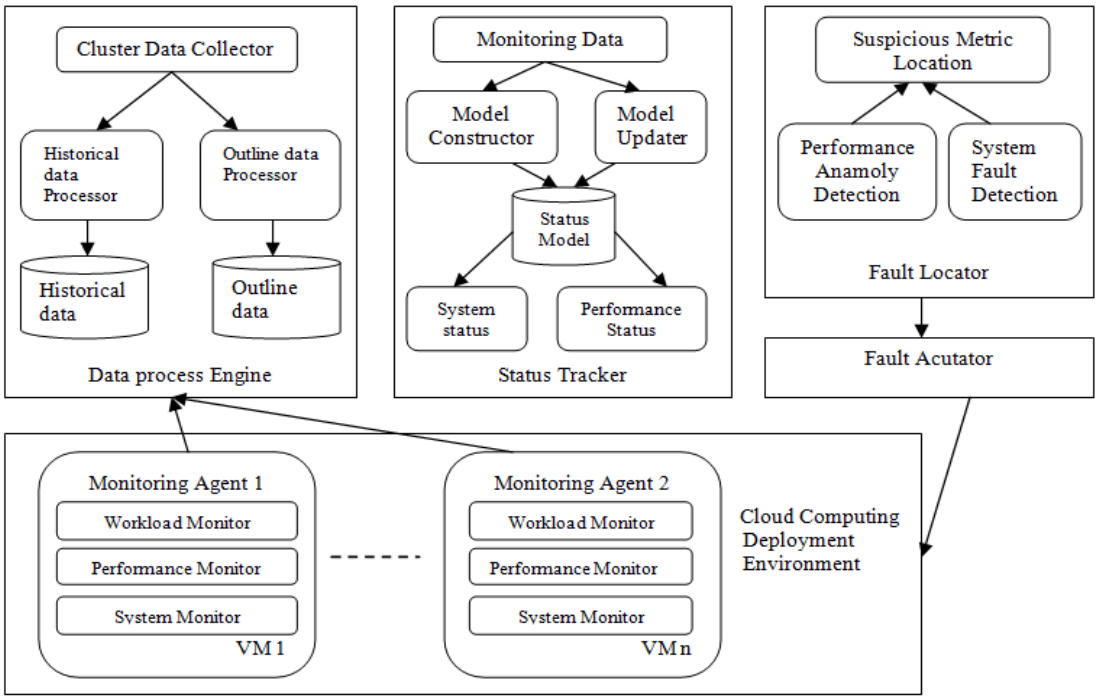

Figure 1: Fault detection /diagnosis Architecture

4.1 Monitoring Agent

A monitoring agents collect monitoring data from various layers of applications (e.g., host metrics, VM metrics, performance metrics, and workloads), and send them to the data processing engine for preprocessing. We can use the interfaces provided by operation systems and virtualization platforms to monitor the system-level metrics. The monitor agent can be any third party software which will monitor the application over cloud and provide us the logs which will contain the measuring parameters like CPU utilization, Memory utilization etc.

4.2 Data Processing Engine

The data processing engine processes the monitoring data online, and sends the preprocessed data to the status tracker for further analysis. In short it reads the data given by monitoring agent, preprocesses it, removes duplicate entry and make it ready for next stage. The engine includes two components: 1) a cluster data collector and 2) data processors. The former collects monitoring data from monitoring agents deployed on different servers. The latter preprocesses the collected monitoring data by standardizing items, clearing outliers, rectifying faults, and eradicating duplicated data. It extracts useful data required by the fault diagnosis methods from various layers.

4.3 Status Tracker

The status tracker characterizes system status based on statistical models with collected monitoring data. Since workloads influence application performance/resource utilization, we characterize system status by correlating workloads and metrics with CCA. Thus, we just track the correlation coefficients to determine whether the system is healthy.

4.4 Fault Locator

The fault locator detects faults online by tracking the change of the characterized system status, and then locates the suspicious metrics related to the faults by comparing the multidimensional monitoring data collected before and after the faults are detected. In this paper, we detect the abrupt changes of coefficients with EWMA control charts which do not require domain knowledge. Then, we adopt a feature selection method combining ReliefF and SVM-RFE to locate abnormal metrics by analyzing the variation of multiple metrics automatically.

4.5 Fault Actuator

The fault actuator can use VM technologies to mitigate the consequences of faults through the adjustment of the resource allocation in the virtualized cloud computing platform. Once we know the root cause and its resolution, system will automatically resolve this and bring application to healthy status.

5. Conclusion

In this paper, we present an automatic fault diagnosis framework for Web applications in cloud computing. We propose an online incremental clustering method to recognize access behavior patterns, and then use CCA to model the correlations between the workloads and the metrics related to the application performance/resource utilization in each specific access behavior pattern. The system improves the accuracy of fault diagnosis in the dynamic environment of cloud computing. Finally, fault diagnosis framework detects and locates faults without domain knowledge, which is suitable for managing large-scale cloud computing systems.

References

[1] D. Oppenheimer, A. Gananpathi, and D.A. Patterson, "Why do Internet services fail, and what can be done about it?" in *Proc. 4th Symp. Internet Technol. Syst.*, Seattle, WA, USA, 2003, pp. 1–16.
[2] G. Jiang, H. Chen, and K. Yoshihira, "Modeling and tracking of transaction flow dynamics for fault detection in complex systems," *IEEE Trans. Depend. Secure Comput.*, vol. 3, no. 4, pp. 312–326, Oct./Dec. 2006.
[3] Y. Zhang, Z. Zheng, and M. R. Lyu, "An online performance prediction framework for service-oriented systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 9, pp. 1169–1181, Sep. 2014.
[4] S. Zhang, K. R. Pattipati, H. Zheng, X. Wen, and C. Sankavaram, "Dynamic coupled fault diagnosis with propagation and observation delays," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 6, pp. 1424–1439, Nov. 2013.
[5] H. Chen, G. Jiang, K. Yoshihira, and A. Saxena, "Invariants based failure diagnosis in distributed computing systems," in *Proc. 29th IEEE Symp. Rel. Distrib. Syst.*, New Delhi, India, 2010, pp. 160–166.
[6] M. Y. Chen *et al.*, "Path-based failure and evolution management," in *Proc. 1st Symp. Netw. Syst. Design Implement.*, Berkeley, CA, USA, 2004, pp. 23–36.
[7] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier, "Using magpie for request extraction and workload modelling," in *Proc. 6th Int. Symp. Oper. Syst. Design Implement.*, Berkeley, CA, USA, 2004, pp. 18–31.
[8] H. Chen, G. Jiang, C. Ungureanu, and K. Yoshihira, "Failure detection and localization in component based systems by online tracking," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Chicago, IL, USA, 2005, pp. 750–755.
[9] G. Jiang, H. Chen, and K. Yoshihira, "Modeling and tracking of transaction flow dynamics for fault detection in complex systems," *IEEE Trans. Depend. Secure Comput.*, vol. 3, no. 4, pp. 312–326, Oct./Dec. 2006.
[10] Y. Tan, H. Nguyen, X. Gu, C. Venkatramani, and D. Rajan, "PREPARE: Predictive performance anomaly prevention for virtualized cloud systems," in *Proc. 32nd Int. Conf. Distrib. Comput. Syst.*, Macau, China, 2012, pp. 285–294.

Author Profile

Kshitija Nandgaonkar received B.Tech. degree in Information Technology in 2013 from Government College of Engineering Amravati(An Autonomous Institute of Government of Maharashtra) and pursuing M.E. from RMDSSOE, Warje, Pune.

Swarupa Kamble is working with RMDSSOE, Warje, Pune as an Assistant Professor. She has experience of 5 yrs in the field of teaching and research and her research interests are Image Processing and Data Mining.