34-F

More Reliable Cloud Systems – A System Accounting Approach

Zhengping Wu[1], Nailu Chu[1], Peng Su[2]

[1]Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT [2]Cisco Systems, Inc., San Jose, CA

Abstract

Cloud computing has attracted more and more attention and has been used in more and more applications in many different fields. As we know, cloud computing delivers computing as a service. Its adoption widely depends upon the reliability of its systems. However, self-monitoring and adaptation are overlooked by most system designers and architects. Extant system accounting functionalities mainly focus on recent error causes, which is insufficient for long-term history analysis and event prediction. Our approach not only analyzes system events from both service consumers and providers, but also provides a layered composable system accounting framework in cloud computing environments. By self-monitoring, history analysis, event prediction and recovery, a new refined quality of reliability (QoR) for cloud computing is essentially provided. An implementation of this framework in an education services environment confirms the advantages over extant system accounting systems.

System Architecture

One of the major reasons to use cloud computing is its reliability. Therefore, QoR of cloud systems is one of the crucial factors for a trustworthy cloud computing provider. Maslow's hierarchy of needs is a good reference through which we can recast the basic reliability requirements, because services are eventually designed to satisfy human needs. Based on the human needs hierarchy (Figure 1), four levels of QoR requirements for cloud systems can be summarized as: Service Existence, Service Availability, Service Capability and Usability, as well as Service Self-Healing (Figure 2).

Figure 1. Maslow's Hierarchy of Needs

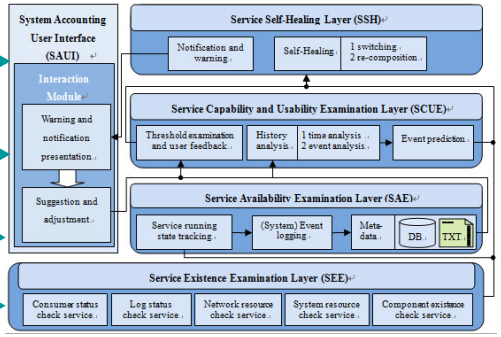

Figure 2. System Accounting Architecture

History Analysis Module

Detection of events (with certain log patterns) and making a comparison between occurrence statistics become prerequisites for event prediction. We introduce a pipelined approach called Pipelining Aho-Corasick (P-AC) that can reduce more cross links than deterministic finite automaton (DFA).

For example, the states of an event level is the set {32, 432, 542, 543}, and Figure 3-a shows every forward edge by a solid line and each cross link by a dashed line. Whereas, by employing a pipelined solution, the cross links in traditional DFA are reduced and the DFA can form a more efficient tree structure, denoted by Figure 3-b. Because, instead of moving to another state according to current input, event logs are stored in corresponding pipelines that are ready to match and check for the next state.

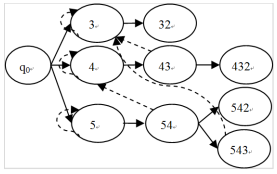

Figure 3-a.
DFA State Graph

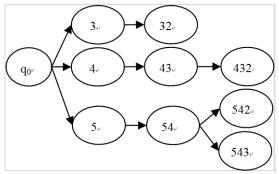

Figure 3-b.
Reduced State Graph Maintained by Pipeline System

Event Prediction Module

In this module, every service's performance is quantified, so that a novel algorithm based on probability dependence can provide more accurate event prediction in cloud computing environment. Based on Naïve Bayes (NB) classifier, an improved model of Averaged One-Dependence Estimators (AODE) is employed in our approach. The probability calculation for any attribute x_i is

$$P(y \mid x) = \frac{\sum_{i:1\le i\le n} P(y \mid x_i)\prod_{j=1}^{k} P(y, x_j \mid x_i)}{\sum_{y'\in Y}\sum_{i:1\le i\le n} P(y' \mid x_i)\prod_{j=1}^{k} P(y', x_j \mid x_i)}$$

$$x \in (X \mid x_1, x_2, \ldots, x_i, \ldots x_n) \quad y \in (Y \mid y_0, y_1)$$

X denotes abnormal event and Y represents irregularities. Due to some unpredictable events leading to Y in a cloud system, such as human interference, power outrage, etc., to calculate the dependency among x_i and eliminate the uncertainty that Y leads to X, we can use $x \in (X \mid x_1, x_2, \ldots, x_i, \ldots x_n)$ and $y \in (Y \mid y_0, y_1)$ to represent corresponding log patterns (pattern length is k). For each occurred y, the averaged probability of certain irregularity can be calculated by this equation.

Simulation

To test the performance for the availability of a cloud system using our system accounting approach, a 200-day experiment has been conducted. A comparison of the data between with and without our system accounting can show our framework's effectiveness.

In Figure 4, x axle represents the number of days; the availability are demonstrated by y axle. As a result, the cloud system with our framework performs approximately 10% better than the one without it and it is more stable in this figure.

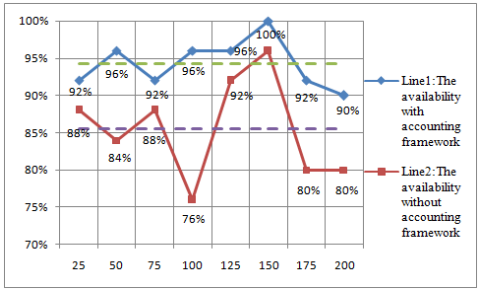

Figure 4. The comparison of system availability between with and without our system accounting approach

Conclusion

Following the requirement definition of QoR for cloud systems, our approach is able to not only monitor the existence of services but also perform history analysis and event prediction. With enough training time and experience accumulation, it will help both providers and consumers achieve better performance in term of reliability.