Gleaming the Cloud in the Virtualization layer

Janani.V.D, M.E.
Computer Science and Engineering
Vels University

Hari Baabu.V, Sajeel ram.A
Assistant Professor
Computer Science and Engineering
Vels University

Abstract—Cloud Computing is a powerful and flexible software environment, which delegates the material's management and in which users pay as they go. Rather than operating their own data centers, today cloud users run their applications on the remote cloud infrastructures that are owned and managed by cloud providers. Cloud users create virtual machine instances to run their specific application logic without knowing the underlying physical infrastructure. On the other side, cloud provider manage and operate their cloud infrastructures without knowing their customers’ applications. Due to the disjoint ownership of applications and infrastructures, if a problem occurs, there is no visibility for either cloud users or providers to understand the whole context of the incident and solve it quickly. To this end, we propose software solution Cloud Vision, to provide some visibility through the middle virtualization layer for both cloud users and providers to address their problems quickly. Cloud Vision automatically tracks each VM instance’s configuration status and maintains their life-cycle configuration record in configuration management database(CMDB). When the user reports a problem, our algorithm automatically analyze CMDB to probabilistic determine the root cause and invoke a recovery process by interacting with the cloud user.

Index Terms—Cloud Vision; Configuration management; Virtualization layer; VM; Troubleshooting;

I. INTRODUCTION

Cloud computing provides a revolutionary new computing paradigm for deploying enterprise applications and Internet services. There are many evolving cloud computing models, including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service(PaaS), and Software- as-a-Service(SaaS), which provide IT infrastructures, platforms, solution stacks, and software applications from the cloud respectively [1]. Cloud infrastructures often employ a virtualization layer [8] to ensure resource isolation and abstraction. They are designed to provide restricted visibility to both users and IaaS providers. Users of VirtualMachine (VM) instances [7] are blocked from looking down into the infrastructure layer since multiple tenants might share the same physical machine. Similarly IaaS providers such as Amazon’s Elastic Computing Cloud (EC2) [2] are not allowed to look inside the running VM instances [7] because of tenants’ business confidentiality and privacy.

In a local data center, operators usually need the context information from both applications and infrastructures for effective problem determination. In commercial cloud environments, the decoupled ownership of applications and infrastructures introduces new challenges in system management. When users encounter problems in their VMs, they have little visibility on the infrastructure layer for root cause determination except trial-and-error troubleshooting [9]. It is not even clear who is responsible for fixing such problems. To support end users, IaaS providers often run an online support forum (e.g., Amazon EC2 online forum [3]) where users can report their problems and seek solutions. Cloud operators often try to solve those problems by manually investigating the system settings of users’ VMs. Since operators do not know tenants’ specific applications, it is even hard for them to solve problems only based on users’ problem reports.

Most of VM problems [7] in a cloud infrastructure (except those problems caused by applications running inside VM instances) can be correlated to their configuration events. Examples of such configuration events include incorrect update of Access Control List (ACL) in hypervisors, blocking of access ports inside running VM instances, connection of virtual devices to unsupported OSes, migration of instances to a slow machine, incorrect allocation of memory size, connection loss with underlying block storage and so on. For example, on average 28 problems were reported daily to Amazon EC2 online forum in July 2010, complaining about VM unavailability, VM hang, Elastic Block Storage (EBS) [2] [7] crash or loose connectivity, performance problem and so on. Problems can occur at the time when an instance is first created from its source image or during running time if its configurations are
changed by some events.

We make two intuitive observations about cloud environments. First, multiple VM instances created from the same image source are likely to have similar characteristics in configurations. Second, some configuration attributes have higher cardinality than others and hence they are less sensitive to configuration changes. Based on these, we present a solution (called CloudVision) for cloud providers to automate reasoning and troubleshooting [9] on user reported problems under the restricted visibility. CloudVision runs in the infrastructure layer and tracks each VM instance as a standard object running in that infrastructure. In summary, we have made the following contributions:

**Representation:** CloudVision develops a monitoring mechanism at the infrastructure layer to automatically capture the event history of all running VM instances in cloud environments [7]. Such event data is further structured and stored in a Configuration Management Database (CMDB).

**Problem Reasoning:** When users report a problem, our solution fetches the related information from the CMDB and further identifies a list of suspect events. We then apply statistical analysis to rank these events based on their sensitivity to configuration changes.

**Interactive Troubleshooting:** Following the list of ranked events, CloudVision takes predefined actions to check and solve problems. For each specific event, operators define the predicates to check whether this event is the root cause as well as the actions to remediate the problem.

### III. CLOUD VISION SYSTEM ARCHITECTURE

The system architecture of CloudVision is shown in Figure 1 and it consists of three major components: Monitoring Agent, Cloud CMDB and Analysis Engine.

![CloudVision Architecture](image)

**Figure 1: CloudVision Architecture**

**Monitoring Agent:** The Monitoring Agent monitors and tracks the configuration attributes of VM instances and infrastructures continuously. We assume that all PM clocks are loosely synchronized. The agent sits in the hypervisor layer so that it can be rolled out through the cloud infrastructures as a standard management component. Section IV-A discusses the details of data collection.

**Cloud CMDB:** CMDB Manager structures and stores configuration data sent by the monitoring agent into a database called Configuration Management Database (CMDB). CMDB is often used for asset and configuration management in large data centers. In this paper, we use a CMDB to track the life-cycle configuration information of VM instances. It is a relational database and each row includes a configuration change along with its timestamp and health status (sick or healthy). Initially all rows are marked as healthy. If a configuration change is found to be the cause of a reported problem, the
row containing this change is marked as sick in the database during the troubleshooting step [9]. In Section IV-B, we discuss how CMDB is designed and populated in cloud infrastructures.

**Analysis Engine:** After users encounter a problem with their VM instances [7], they report the problem to CloudVision via a web interface. The Analysis Engine fetches the data related to the reported VM from the CMDB and compares its current configuration set with its historical records to determine a list of suspect events. Then the engine employs statistical analysis to rank the list of suspect events and uses predefined predicates and actions to check these events and resolve the problem. Not all problems can be automatically resolved by CloudVision since some problems are beyond the management responsibility of cloud providers. However, the suspected events are reported to end users with an interactive troubleshooting process [9] and its design details given in Section IV-C.

**IV. CLOUD VISION DESIGN AND IMPLEMENTATION**

**A. DataCollection**
Cloud Vision runs an event-based monitoring agent at the hypervisor layer of each physical machine. The agent collects system configuration data shell scripts. The shell scripts use virsh management interface library to collect values of configuration attributes of running VM instances and it also collects IP addresses of running guests by looking into the system. Collected information are represented using an XML format. Configuration data is collected periodically but it is not sent to the CMDB Manager unless there is a change in the configuration data compared to the previous one, which indicates new events. Hence the data collection is event-based to reduce network overhead.

**B. Populate CMDB**
The central CMDB Manager node runs a program that collects event-based configuration data from monitoring agents. It parses collected XML messages and uses a relational database to structure and store all configuration data along with the reported timestamp. Each row in the database indicates at least a single change in the set of configuration attributes and it is also associated with a status tag that defines whether the configuration is sick or healthy. Initially all entries are set to be healthy and a configuration is changed to be sick only if it is proved to be faulty during the troubleshooting process.

**C. CMDB Analysis**
When a problem is reported, the analysis engine analyzes the related CMDB records of the reported VM instance [7] to determine the suspect events and then tries to check the root cause and solve the problem. The whole process is divided into three steps: Identifying Suspect Events, Event Ranking and finally Interactive Troubleshooting.

1) **Identifying Suspect Events:** when a problem of a VM instance occurs, it is most likely caused by the recent events of that VM instance. As the CMDB maintains the complete event history of VM instances, the analysis engine can determine the list of suspect events from the CMDB. Each event is associated with a change in the configuration attribute called suspect attribute. CloudVision identifies the changes of the reported instance by comparing its current configuration to the latest known stable configuration. Any configuration that persists longer than a specific duration of time is considered as a stable configuration. Unstable configurations may arise when users try to fix problems by trial and error so that every configuration change only lasts for a very short period of time.

2) **Event Ranking:** After determining the set of suspect events, CloudVision ranks them based on how likely an event could be the problem root cause. As the set of suspect events can be large, we reduce the problem resolution time by checking the events in their ranked order. An event is highly suspected if it is very rare to occur. To measure this, we define a sensitivity metric and rank the set of suspect events according to their sensitivity, i.e., the sensitivity of an attribute to change. We define two types of event sensitivity: local sensitivity and global sensitivity. The local sensitivity defines the probability of an event’s occurrence using the event history of the instances originated from the same image source. Meantime the global sensitivity defines the probability of an event’s occurrence using the event history of all VM instances regardless of their image sources.

3) **Troubleshooting:** After determining and ranking suspect events, CloudVision performs troubleshooting actions, which include two major steps: Check and Solve. For each suspect event, the check step checks whether it is the true root cause (or culprit event) of the problem. The solve step tries to solve the problem automatically if the solution is within the management responsibility of the cloud provider, otherwise CloudVision recommends the solutions to the cloud users. To identify the root cause, Cloud Vision uses a series of predefined predicates to check each suspect event, starting from the top of the ranked suspect events. The suspect events are
first classified into one or more problem classes. Actions are defined for each individual problem class and also include a series of predicates to check the existence of the corresponding problem.

**Figure 2: Check-and-solve flow chart**

The check-and-solve troubleshooting process is done interactively with users. After each solve step, users are asked to provide feedbacks. If the problem is solved, the troubleshooting process ends; otherwise the check-and-solve steps continue for the next suspected event. This process is very similar to the troubleshooting support for desktop printing or networking in Windows OS. Sometimes the check-and-solve steps need to shutdown or restart a instances to solve the problem, which has to get user’s confirmation. If some users want to run their instances without any interruption, Cloud Vision sends the solution back to the users for their own decision. After a problem is reported, Cloud Vision automatically invokes problem reasoning and uses the same web interface to interact with users and acquire their confirmation and permission for actions. Figure 2 shows the complete flow chart of our check-and-solve troubleshooting algorithm [9].

**V. RELATED WORK**

Much work has applied statistical learning methods to detect and diagnose problems in large-scale computer systems. Cohen et al., [4] used a tree-augmented naive (TAN) bayesian network to learn the probabilistic relationship between SLA violations and system resource usages. They used this learned bayesian network to identify performance bottlenecks. Bodik et al. [5] proposed a “Fingerprint” approach, which identifies the problem with performance fingerprinting. There also exist some literature on configuration management. Whitaker et al. [6] developed a tool named Chronus to automatically search for a configuration state change indicating a failure. It stores local system configurations at each checkpoint along time. When a problem occurs, Chronus takes user-provided probes to identify the working and non-working configuration state. This approach is quite similar to our check and solve step.

**VI. CONCLUSION**

In this paper, we present Cloud Vision, a novel solution for automated problem troubleshooting in cloud environments. Cloud Vision monitors and tracks the configuration attributes of VM instances [7] and infrastructure, and uses the historical event records to determine the root cause of problematic VM instances, and further provides a check-and-solve troubleshooting process to resolve user reported problems automatically. In our future work, we plan to monitor more configuration attributes from cloud infrastructure and design new predicates and actions to cover more problem classes.

**REFERENCES**