Enhancing Cloud Availability using Artificial Intelligent Operations (AIOps)

Parthasarathy PD, Research Scholar, BITS Pilani, India, pdparthasarathy.03@gmail.com

Dr. Vinod Vijayakumaran, Professor, BITS Pilani, India, vinod.vijayakumaran@gmail.com

Abstract - Every cloud provider wishes to provide 99.9999% availability for the systems provisioned and operated by them for the customer i.e. may it be SaaS or PaaS or IaaS model, the availability of the system must be greater than 99.9999%. It becomes vital for the provider to monitor the systems and take proactive measures to reduce the downtime. In an ideal scenario, the support colleagues (24*7 technical support) must be aware of the on-going issues in the production systems before it is raised as an incident by the customer. But currently, there is no effective alert monitoring solutions for the same. The proposed solution presented in this paper is to have a central alert monitoring tool for all cloud solutions offered by the cloud provider. The central alert monitoring tool is intelligent enough to find if the cloud system is in danger and performs a self healing and brings the system out of threat and helps to achieve the availability goal. The AI powered intelligent monitoring tool uses machine learning techniques to decide if the system needs a self healing. This paper acts as a guide to cloud providers and cloud practitioners.

Keywords —Artificial Intelligence, cloud availability, time series database (TSDB), machine learning (ML), Host Agent (HA), self healing.

I. INTRODUCTION

Availability refers to the uptime of a system, a network of systems, hardware, and software that collectively provide a service during its usage [1]. Traditionally the availability of these has been limited to local installations of hardware and software resources which businesses and consumers deployed and maintained. With the advent of cloud services, there is a considerable shift of these resources into the cloud. While cloud computing presents some cost-effective benefits for the consumers and businesses, it is also extremely important for the cloud service providers to offer environments that are highly available. Regardless of the size of an organization prolonged downtime of the service (these days a few minutes) might be disastrous to its business, customer loyalty and brand value.

Ideally, the cloud service providers are expected to offer much more robust cloud infrastructures to ensure that their customers are presented with an environment that is highly available. This is true for the public, private and hybrid models of the cloud. But, the situation with most cloud providers is that they are not aware of the on-goings in the infrastructure provisioned by them for the customer. The cloud provider is aware of an issue in the customer system only when the customer raises an incident in the cloud providers support tool. This is a big threat to the cloud providers brand value and hence it becomes vital for the providers to monitor the infrastructure provisioned by them and take proactive measures to reduce the downtime.

In an ideal scenario, the support colleagues (24*7 technical support) must be aware of the on-going issues in the production systems before it is raised as an incident by the customer. Currently, there is no effective alert monitoring solutions for the same. This paper proposes an effective alert monitoring solution which can act as a reference guide for cloud providers and cloud computing practitioners.

II. CURRENT APPROACH AND ISSUES

Current Approach

In the current scenario, there is no alert monitoring in place and the cloud providers get to know about a downtime or an issue only a customer reports the same via the cloud providers support channel after which the analysis begins. Also, to increase the availability the most common solutions adopted by providers is heartbeat/echo, High Availability (HA) and Disaster Recovery (DR) mechanisms. The HA/DR techniques introduce redundancy at the system, database, application level so that when there the primary system is down, there is an immediate switch to the standby systems. They come with various flavors such as Active/Passive and Active/Active systems. In heartbeat/echo, a component issues a ping and expects to receive back an echo, within a predefined time, from the component under scrutiny. If the echo is not received within the predefined time, it is assumed to have failed.

Issues

As discussed in the above, echo/heartbeat technique is limited only to detecting a failure and does not focus on

 DOI : 10.35291/2454-9150.2019.0042

intricate details of components involved. A ping check also helps only to know the overall availability of the system. There is no intricate precautious monitoring at various levels in these approaches. Meanwhile, the other approaches like HA/DR and fail-over are **reactive** approaches to increase availability. Hence, the proposed approached in this paper tries to overcome the drawbacks of these by having a predictive-precautious monitoring at each level of the system (OS, DB, application) in a **proactive** manner.

III. KEY TERMS ELABORATED

- **Availability**

Availability is defined as the percentage of time that a system is available to perform its required function(s). It is measured in a variety of ways, but it is principally a function of downtime [2]. Availability can be used to describe a component or system, but it is most useful when describing the nature of a system of components working together. Because it is a fraction of time spent in the "available" state, the value can never exceed the bounds of 0<A<1. Thus, availability will most often be written as a decimal, as in 0.99999, as a percentage, as in 99.999%. Availability is influenced by the time demand made by preventive and corrective maintenance measure.

Availability (A) is measured by: A = `MTBF/(MTBF+MTTR)`

where, *MTBF:* Refers to the amount of time that elapses between one failure and the next and *MTTR:* Refers to the amount of time required to repair a system and bring it back to its fully functional state.

- **Time Series Database [TSDB]**

A time series database is a software system that is optimized for handling time series data, array of numbers indexed by time [3]. Software with complex logic or business rules and high transaction volume for time series data may not be practical with traditional relational database management systems. Flat file databases are not a viable option either. In such situations, a time series database comes into play. It is a specialized database for storing and retrieving time series in an efficient and optimized way. Some common time series databases are RRDTool, Graphite, InfluxDB, OpenTSDB, KairosDB, SAP HANA and many more!

In this use case, metric data is coming from many systems/infrastructures every few seconds and hence a timeseries database is needed.

- **Machine Learning**

ML is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" with data, without being explicitly programmed [4]. The machine is trained using large amounts of data and this acts as the input often referred to as training data. The training data is added to a training algorithm which analyses the data and recognizes patterns, the outcome of this algorithm is knowledge [5].

- **Need for Machine Learning**

In this use case, metric data is coming from many systems/infrastructures every few seconds. This data may contain noise and outliers. The incoming data cannot be directly compared with the metric threshold data to conclude if there is an issue. ML algorithms can help in getting rid of this and help us cluster the data enabling us to do a better analysis. The Decision tree are also useful for the predictive modeling and self-healing.

- **Host Agent**

Host Agent is an agent which can accomplish several life-cycle management tasks, such as system provisioning, instance control and provisioning, installation manager, operating system and database control. Host agent in installed automatically during the installation of new systems/infrastructures. The host agent is upgraded automatically as part of the system, when the system is patched or upgraded. Hence, most cloud providers have such an agent for the installation and management of systems with their own set of functionalities and name. Example: SAP Host Agent (SHA), NetApp Host Agent (NHA) and so on. Hence, every cloud provider provisioned infrastructure/system has an agent in the system via which the system was provisioned and can be managed. The idea is to leverage the capabilities of this agent for constant monitoring. The approach is to define relevant metrics/monitoring parameters and their thresholds for each type of system (ERP, CRM, CMS, MAP, PIM, AMS etc.) at the operating system level, database level and application level. The HA collects the real-time value of these metrics in said intervals (ideally in milliseconds) and pushes it into a time series database.

- **Artificial Intelligence**

Artificial intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using rules to reach approximate or definite conclusions) and self-correction.

- **Self-Healing**

In software systems, the self-healing term describes any application, service, or a system that can discover that it is not working correctly and, without any human intervention, make the necessary changes to restore itself to the normal or designed state. Self-healing is about making the system capable of making its decisions by continually checking and optimizing its state and automatically adapting to changing conditions. The goal is to make fault tolerant and responsive

 DOI : 10.35291/2454-9150.2019.0042

system capable of responding to changes in demand and recuperation from failures. The three levels of self-healing are at

1) Application level
2) System level
3) Hardware level

IV. PROPOSED APPROACH

The proposed solution is to have central alert monitoring application which helps in predictive, proactive monitoring of the customer systems and performs self-healing when needed without the intervention of human beings.

It monitors the landscape of customer systems based on metric values that are stored into a TSDB and creates incidents in the providers support tool when the metric value goes beyond the configured metric threshold values. The support tool is AI powered meaning has the intelligence to self-heal the issue. If that step fails, it sends a notification to the human Support desk.

Stage 1

The basic architecture of the proposed solution as in stage 1 is as shown in Fig 1. The steps 1 through 6 in the diagram is as explained below:

Step 01: The cloud provider uses the system provisioning tool to setup the customers systems. Once the system is setup and is live, it is delivered to the customer. The provisioning data such as the type of system, different metrics that are relevant for such a kind of system, the thresholds of each metric etc. are stored in the Provisioning and Metric Details Database of the System Provisioning tool.

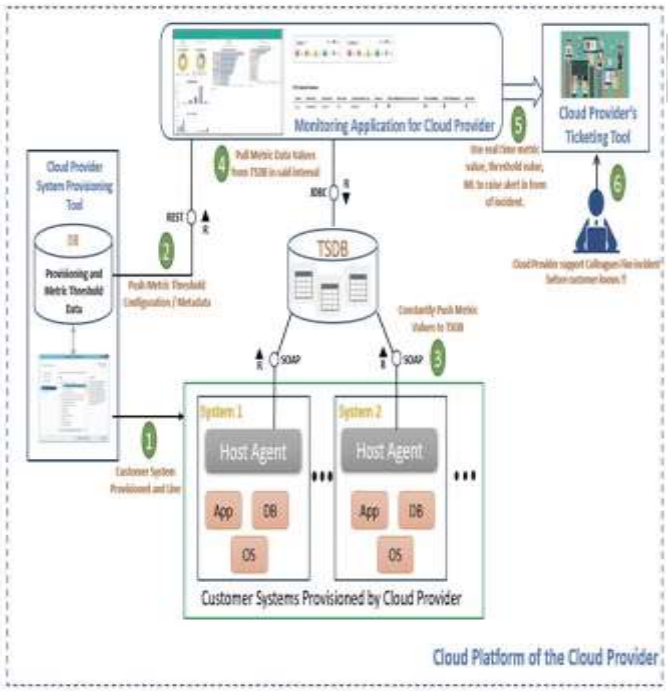

Fig. 1. A Basic Architecture of the Proposed Solution Stage 1

Step 02: The provisioning/system data along with the metrics data of the newly provisioned systems are pushed into the monitoring application. This data is vital for the monitoring application as this acts as the metadata and the monitoring application knows which systems to monitor, what are the metrics and their thresholds respectively.

Step 03: The moment the customer's system goes live, the host agents in the systems get activated and start pushing real-time metric values to the TSDB on said intervals. The metrics are spread across all levels namely, application level, database level, and operating system level. The intervals are configured when the system is setup in the system provisioning tool. Each metric can have its own time interval of pushing data, example: the hostagent may push the metric value of a critical metric 'X' every 5 milliseconds while it may push the metric value of a non-critical metric 'Y' every 5 seconds into the TSDB.

Some of the metrics are CPU utilization, load, disk space usage, disk space available percent, disk throughput, disk latency, disk utilization, page size, number of active pages, application specific metrics like number of active users and so on. This activity of real-time metric values getting pushed into the TSDB never stops unless explicitly told to do so.

Step 04: Once, the Host agent's start pushing real-time metric values into the database, the monitoring application starts its action. The monitoring application polls the TSDB every 1 second to get the collected values in the past 1 second. The application does a comparison of the incoming metric values and the metric's threshold and performs the necessary action. Hence, after the application and the customer systems are live, the step 03, step 04 happens iteratively in parallel.

Step 05: If the incoming metric values are beyond the accepted threshold, the application uses ML techniques to check if this could a potential threat causing an issue in the system or is an outlier or a noise. In case of an outlier or a noise, the application does not take an action but uses this data for its learning. In case the ML algorithm finds that it is a valid case and the metric value is going beyond threshold, the application automatically creates an incident in the support tool of the Cloud provider which is AI powered, which we will discuss in Stage 2 of the proposed approach. This is done using the decision tree algorithm as shown in Fig. 2.

 DOI : 10.35291/2454-9150.2019.0042

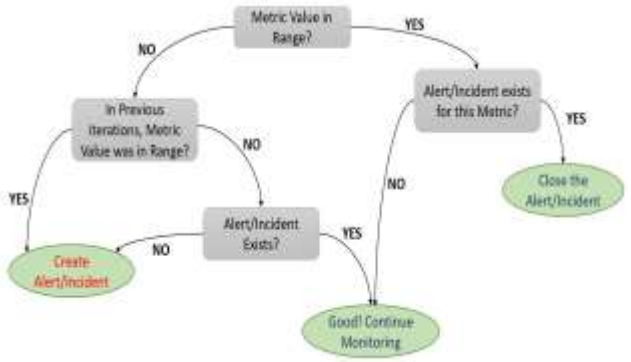

Fig. 2. Machine Learning – Decision Tree in Action to Determine the action based on Metric value

The application also keeps an eye on the raised incidents to check if the incidents raised still holds good i.e. it constantly monitors the incoming metric values with the threshold values again as usual and say suppose in the next few iterations, if the metric for which an incident was raised is back to normal (if the metric in below threshold constantly for say 1 minute) then the application is intelligent enough to close the raised incident automatically.

Stage 2

This is the stage where the Cloud Providers Support tool comes into play. This tool is powered by AI and has the capability to perform self-healing based on adaptive learning algorithms, incremental learning and transfer learning. Whenever step 05 in stage 1 has raised an incident, the incident reaches this AI powered tool which using its AI performs self-healing.

The self-healing that the tool is capable are only at the application level and system level. An example of this could be: If an incident was raised for abnormal CPU usage, the AI powered support tool will check the system, close unwanted processes etc. The self-healing is performed carefully as it should not have any after effects due to its action. If the AI powered tool fails to perform a self-healing or does not know an action (in case of a new issue which it could not identify using ML techniques) or in case of hardware level issues, the tool simply forwards the incident created in Step 05, Stage 1 to the manual support which is the 24 * 7 support staffs who can start analyses and resolve them.

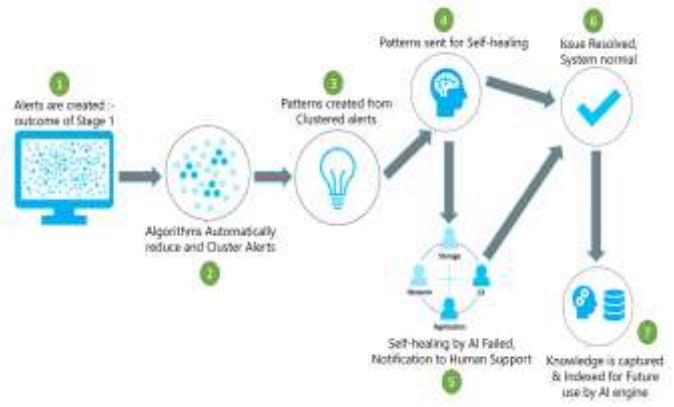

Fig.3. Process of AI Powered Support Tool

The AI powered tool raises/forwards only valid issues that it could not resolve using its intelligence and closes the incidents which are no longer valid (as explained in the previous step). In this manner, the cloud provider is aware of the customer systems and can resolve issues if any to increase the availability of the customer systems. The customer is not even aware of such an issue which is ongoing in their landscape and can be assured of the availability of their systems.

CID: Number	SID: Number	Metric Path: String	Metric Threshold: Base Value	Metric Threshold: Limit Value	Metric Value: Numeric	Issue Flag: bool	Start_TS: BIGINT	STOP_TS: BIGINT

As shown in Fig.3. there are seven steps that the AI powered support tool does. The role of each step is briefly described below:

Step 01: The outcome of Stage 1, the incidents have arrived in the AI powered support tool of the cloud provider.

Step 02: The incidents are clustered using K-Means, K-neighbors etc. and other ML techniques

Step 03: Patterns are identified, and the existing knowledge base is checked for self-healing mechanisms.

Step 04: Sent to the AI Engine which performs the self-healing procedure/mechanism. If the healing was successful, goto step 6.

Step 05: If the AI Engine failed to perform the self-healing, then the human support (Development support) staffs are intimated so that they can check it manually and fix the issue. Also, the knowledgebase is enhanced with this knowledge of issue and fix.

Step 06: The issue is resolved, and the system is behaving normally. Also, the customer was not even aware of the ongoing issue and its fix.

Step 07: The knowledge is captured & feed to the knowledgebase so that AI engine can use it in future.

V. IMPLEMENTATION AND RESULTS

The steps as explained in the previous section were implemented and the schema of the table currently used for this implementation in TSDB is as shown below:

Where; CID is the customer ID, SID is the unique system ID which is under scrutiny, MetricPath is the absolute path of the metric along with the metric name, metric threshold base value and limit value are the threshold values of the corresponding metric, metric value indicating the real-time metric value that is pushed from the systems Hostagent and the issue flag representing a potential threat and which is filled as a result of applying ML techniques as discussed in the previous section, the last two attributes are the timestamp when the metric was inserted. The k-means

 DOI : 10.35291/2454-9150.2019.0042

algorithm is used to identify the noise in our implementation. After implementing the solution in our premise, the incidents and outage dump of last two quarters where analyzed and the outcome is as shown in Fig. 4.

As seen in the Fig. 4 (which shows the comparative analysis of the performance of the existing system and the proposed system), the number of incidents raised by customers in the support tool of the cloud provider is quite high before the monitoring application is introduced (The 2 incidents in that case is raised by operations team). After the introduction of the monitoring application and AI powered support tool, we observe that there is considerable number of internal incidents raised automatically by the tool and there are minimal incidents by the customer (only 1).

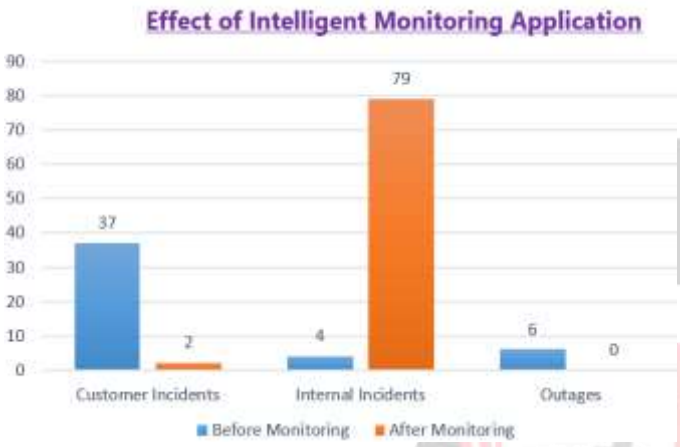

Fig. 4. Effect of Intelligent Monitoring Application

(Comparative Analysis of existing and proposed approach).

We can also observe that the number of actual outages is narrowed down to 0 as a potential issue is raised as an internal incident and before it turns into a threat, it is resolved by the AI Engine or the support staff. Also, among the 79 automatic incidents create as the outcome of Stage 1, 56 of them were solved by the AI engine using its ML and self-healing procedures and 23 by human support staffs as shown in Fig.5. Also, by incremental learning and knowledgebase updation, the 29% of human intervention can be brought down to less than 10%.

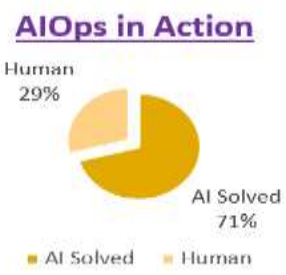

Fig.5. AIOps in Action

Hence, after the introduction of the intelligent monitoring application, the customer can be assured of the availability of their systems and the cloud provider and keep the promise of 99.9999% availability of the systems provisioned and maintained by them.

VI. Challenges

- **Self-Monitoring**

There might be a case when the TSDB is out of memory, the intelligent monitoring tool itself is not performing as expected. To avoid this case, we need to have self-monitoring capabilities from the intelligent monitoring tool itself.

- **Defining the Time interval for Data Push**

If the time intervals for pushing data from Host agent to TSDB and the time intervals at which intelligent application polls TSDB is kept minimal then the monitoring is more precise and efficient but this might become an overhead as there is constant load on the monitored systems as they have to push data every few milliseconds and also there would be a network overhead and a higher bandwidth is required. Thus, there must be a tradeoff while deciding the time interval.

- Stability and Reliability of Self-healing procedures and the ML techniques used must be chosen very carefully as choosing the wrong technique would end in negative consequences.

VII. Conclusion

The proposed solution in this paper can be used by cloud providers to increase the availability of the systems that they provision for the customers. Currently the intelligent monitoring application acts as an alert monitoring solution and as a self-healing agent using ML techniques. As seen in the results section, there is a considerable percentage of human intervention even now. The future scope is to reduce this to less than 10% manual intervention. In the manner, the whole availability scenario can be automated without any manual intervention. This manuscript is aimed at guiding and forming a hub for all cloud practitioners in achieving high-availability in the systems that they provision.

VIII. Abbreviations

- AI – Artificial Intelligence
- SaaS – Software as a Service
- PaaS – Platform as a Service
- IaaS – Infrastructure as a Service
- HA – Host Agent
- ERP – Enterprise Resource Planning
- CRM – Customer Relationship Management
- CMS – Content Management System

 DOI : 10.35291/2454-9150.2019.0042

- MAP – Marketing Automation Platform
- MTTF – Mean time to failure
- MTBF – Mean time between Failures

REFERENCES

[1] Network and communication technologies, *Availability of Services in the Era of Cloud Computing*, [Online]

[2] SlideShare, *Implementing the Reliability strategy by Chariton Inao*, [Online]

[3] Wikipedia, "*Time series database*", [Online]

[4] Machine Learning, Wikipedia, [Online]

[5] Alex Smola and S.V.N. Viswanathan "A Taste of Machine Learning" in Introduction to Machine Learning, Cambridge: Cambridge University Press,2008, p.4

[6] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*,7th Ed., McGraw Hill International Edition,2017

[7] Ian Sommerville, *Software Engineering*, 9th ed., Pearson, 2014

[8] Len Bass, Paul Clements, Rick Kazman, *Software Architecture in Practice*, 3rd Edition, Addison Wesley, 2013

[9] Dong Seong Kim, Fumio Machinda, Kishor S. Trivedi, "*Availability modeling and analysis of a virtualised system"*, 2009 15th IEEE Pacific. Rin In.

[10]Host Agent, [Online], Available: nscsap.blogspot.com/2015/09/installing-or-updating-sap-host-agent

[11] Korolev, A (2016) "Using fuzzy models by systems engineers at the stages of system lifecycle" in Proceedings of the First International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'16) (451): 209-220, https://doi.org/10.1007/978-3-319- 33816-3_21

[12] Samsonovich, A. (2015) "Empirical Measure of Learnability: A Tool for Semantic Map Validation" in Procedia Computer Science (71): 265- 270, http: doi.org/10.1016/j.procs.2015.12.223

[13] Kaufman, L. and Rousseeuw, P.J. (1987) "Clustering by means of Medoids" in Statistical Data Analysis Based on the L1–Norm and Related Methods edited by Y. Dodge, North-Holland: 405–416

AUTHOR'S PROFILE

Mr. Parthasarathy PD, has 3+ years of IT Industry experience and currently employed as a full-stack developer at SAP Labs India Pvt. Ltd.
His areas of specialization are on front-end technologies such as MEAN Stack, JavaScript, OOJS, SAP UI5, Fiori Guidelines, ASP.net and backend topics such as SAP ABAP, C# and Java and is a Machine Learning Enthusiast. He has contributed to various open source projects and Innovation topics within SAP. He is also a passionate trainer and coaches' newbies on various technical topics of *Computer Science and technologies*. On an academic front, he is a gold medalist at graduation and post-graduation level and loves applied research and wishes to pursue a PhD soon.

Dr. Vinod Vijayakumaran has 15 years of IT industry experience and currently employed as Technical Project Manager with HANA Enterprise Cloud, Partner Centre of Expertise, at SAP Labs India Pvt. Ltd. He holds doctorate from Karunya University, Coimbatore in the field of Computer Applications. His research area is on processes and this thesis was on a combined software development model incorporating LEAN, Agile and Six Sigma methodologies. Vinod holds a patent from USPO on his tool 'Downtime Calculator'

 DOI : 10.35291/2454-9150.2019.0042