A Proactive Health Monitoring Tool for Software Defined Datacenters

A project report submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

of

Bachelor of Technology

in

Information Technology

by

Nikhil Gupta

Reg. No. 140911466

Under the guidance of

Dr.Raghvendra Achar
Associate Professor
Department of I & CT
Manipal Institute of Technology
Manipal, India

Karthigai Priya Pandian
Sr. Manager
Site Reliability Engineering Team
VMware India
Bangalore, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

June 2018

I dedicate my thesis to my friends and family who have always been a constant source of support and encouragement during the challenges of my whole college life. This project work is also dedicated to my mentors, who have been my source of inspiration and continually provided guidance both morally and technically.

DECLARATION

I hereby declare that this project work entitled **A Proactive Health Monitoring Tool for Software Defined Datacenters** is original and has been carried out by me in VMware Software India Pvt. Ltd., Bangalore and Department of Information and Communication Technology of Manipal Institute of Technology, Manipal, under the guidance of **Priya Pandian**, **Sr. Manager - SDDCaaS Dev Ops**, VMware Software India Pvt. Ltd., Bangalore and **Raghvendra Achar**, **Assistant Professor**, Department of Information and Communication Technology, M. I. T., Manipal. No part of this work has been submitted for the award of a degree or diploma either to this University or to any other Universities.

Place: Bangalore

Date :10-06-18

Nikhil Gupta

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that this project entitled **A Proactive Health Monitoring Tool for Software Defined Datacenters** is a bonafide project work done by Mr. Nikhil Gupta at Manipal Institute of Technology, Manipal, independently under my guidance and supervision for the award of the Degree of Bachelor of Technology in Information Technology.

Dr. Raghvendra Achar
Associate Professor
Department of I & CT
Manipal Institute of Technology
Manipal, India

Dr.Balachandra
Professor & Head
Department of I & CT
Manipal Institute of Technology
Manipal, India

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my mentor Keerthi Kumar and my manager Priya Pandian here at VMware as well as Raghvendra Achar back at college for providing their invaluable guidance, comments and suggestions throughout the course of the project. I would also like to thank Dr. Balachandra, HOD, I&CT Department, Manipal Institute of Technology, Manipal for providing me the opportunity to undergo this project. This project would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

ABSTRACT

In a recent partnership with Amazon, VMware has provided its clients an option to move from on-premise data centres to the public cloud provided by Amazon for added flexibility, in the form of pre-configured Software Defined Data Centres (SDDCs). As a member of the SRE team, one must work on tools meant for internal consumption that help with keeping these systems up and running. Continuous effort must be made to automate as much of the diagnosis and remediation as possible to scale up smoothly as the number of clients increase.

This project particularly focuses on detecting errors and remediating them before they adversely affect an SDDC. A statistical analysis can help deduce information about the health of component systems and help find the root cause of any performance degradation, which can in turn be remediated.

The software that was designed through the period of this internship can efficiently and accurately determine any faults in the SDDCs and also determine which exact component caused the issue. It has been thoroughly tested on production data centres and locally deployed testbeds. Determining the erroneous component that caused the issue immensely helps in narrowing down the range of logs to go through when trying to debug system errors. Once the cause has been identified, it can be mitigated through the use of Remediation and Troubleshooting Service (RTS), which is an internal interface tool that runs custom scripts against deployed SDDCs.

[**Computer systems organization**]: Dependable and fault-tolerant systems and networks – Reliability, Availability, Maintainability and maintenance

[**Computing methodologies**]: Machine learning - Unsupervised learning, Online learning settings

[**Theory of computation**]: Theory and algorithms for application domains – Machine learning theory

Contents

Acknowledgements **iv**

Abstract **v**

List of Figures **x**

Abbreviations **x**

Notations **xi**

1 Introduction **1**

1.1 Present Day Situation 1

1.2 Problem Definition and Motivation 2

1.3 Areas of Work 2

1.3.1 Cloud Computing 2

1.3.2 Software Engineering 3

1.3.3 Data Science 3

1.4 Project Objectives 3

1.5 Target Specification 4

2 Literature Survey **5**

2.1 Virtualization Technology 5

2.2 VMware Products 6

2.2.1 VMware ESXi 6

2.2.2 vCenter Server 7

2.2.3 vSAN 8
2.2.4 vMotion 9
2.3 Statistical Analysis and Machine Learning 9
2.3.1 Control Charts 10
2.3.2 K-Means 12
2.3.3 Kernel Density Estimation 12
2.4 Introduction to the Project Title 15
2.4.1 Reactive Monitoring 15
2.4.2 Proactive Monitoring 15

3 Methodology **17**
3.1 Software Architecture 17
3.1.1 Data Collector 19
3.1.2 The Database 20
3.1.3 POP Reverse Proxy 20
3.1.4 Processing Appliance 21
3.2 Data Analysis and Error Prediction 22
3.2.1 Clustering 22
3.2.2 Kernel Density Estimation 24
3.3 Remediation and Troubleshooting Service 26
3.3.1 RTS Design 27
3.3.2 Remediation procedure through RTS 28
3.3.2.1 Example scenario 31

4 Results **34**

5 Conclusion and Future Work **36**
5.1 Conclusion 36
5.2 Future Work 36

References **38**

ProjectDetail **38**

List of Figures

2.1 VMware ESXi 6
2.2 vCenter Server Appliance hierarchy 7
2.3 Hyperconverged vSAN system 8
2.4 vMotion in action 9
2.5 Types of machine learning 10
2.6 Control chart 11
2.7 Normal distribution in data 11
2.8 K-means in action 13
2.9 Kernel Density Estimation 14
2.10 Bandwidth in KDE 14

3.1 Software Architecture 18
3.2 Layers evaluated from high impact to low impact components 18
3.3 Software Workflow 19
3.4 Reverse Proxy 21
3.5 Repetitive K-Means algorithm 23
3.6 Kernel Density Estimation Flow 25
3.7 RTS Architecture 29
3.8 Log Collection Interface 30
3.9 vSan Scripts 31
3.10 Sample execution output 32
3.11 vSan Runbooks 32
3.12 A runbook to check the overall physical disk health 33

ABBREVIATIONS

API	:	Application Programming Interface
AWS	:	Amazon Web Services
CPU	:	Central Processing Unit
EC2	:	Elastic Compute Cloud
ESX	:	Elastic Sky X
JSON	:	JavaScript Object Notation
KDE	:	Kernel Density Estimation
LCL	:	Lower Control Limit
PoP	:	Point of Presence
PSC	:	Platform Service Controller
REST	:	Representational State Transfer
RTS	:	Remediation and Troubleshooting Service
SAN	:	Storage Area Network
SDDC	:	Software Defined Datacenter
SCP	:	Secure Copy
SRE	:	Site Reliability Engineering
SSH	:	Secure Shell
UCL	:	Upper Control Limit
UI	:	User Interface
VC	:	vCenter
VCSA	:	VMware vCenter Server Virtual Appliance
VMC	:	VMware Cloud
VSI	:	VMware Sys Info

NOTATIONS

μ : Mean

σ : Standard deviation

Chapter 1

Introduction

This chapter briefly describes the project by expanding on a few basic but important points the use case of the project from the companys perspective, the motivation to carry the project out and the objective it focuses on achieving, the consequences that will follow and a timeline of how the project proceeded.

1.1 Present Day Situation

The Site Reliability Engineering team is responsible for the deployment and maintenance of Software Defined Data Centers (SDDCs) supplied to the companys clients. Since the product was launched in the month of December of 2017, the number of clients that the team caters to is relatively small. However, by the end of 2018, the number of clients is expected to skyrocket from a few tens to a thousand.

In the case of a system error, members of the team go through log dumps and try to figure out where the issue originated from. However, investigating a problem after it has occurred is not only stressful, but also inconvenient to clients. If the problem is not resolved within the maintenance limits specified in the SLA, which are as low as 30 minutes per month, it negatively affects the company.

1.2 Problem Definition and Motivation

Since log analysis to find errors is a time-consuming process, manual resolution of erroneous incidents poses a threat of crossing the time limits specified in Service Level Agreements, there is a definite need of a software that can detect system problems beforehand to some degree, by proactively analyzing some critical performance metrics of the systems in an SDDC and reporting any anomalous behavior in the analyzed system components.

Not only should the system predict anomalous behavior, it should also determine the root cause of the anomaly. Reliability engineers can then prevent service outages by looking into the faulty component and fix it as soon as it is detected.

Automation is hence increasingly important over the lifetime of the product since it ensures scalability into a bigger market. It is an effective way to continue services without having to acquire extra human resources.

1.3 Areas of Work

The project spans across a multitude of computer science subjects and uses several state-of-the-art technological tools and concepts. Some of the major concepts that are extensively involved are

1.3.1 Cloud Computing

The main theme of not only the project, but the companys business orientation is cloud computing. At a time when all companies are shifting towards cloud-based data centers to support their business operations in order to reduce costs, VMwares latest projects are naturally aligned to the trend. This project is built for data centers deployed on the cloud exclusively, and an understanding of how the cloud environment works was a must to get started.

1.3.2 Software Engineering

To build a robust software, it is necessary to follow some cardinal guidelines set forth by software engineering principles. Without a proper structure, it becomes increasingly difficult to maintain code as the size of the codebase starts to grow. It was therefore important to make sure that the foundation of the software was flexible enough to accommodate changes and allow for incremental additions to the code.

1.3.3 Data Science

Data is the backbone of any monitoring system; Without the right metrics and logs, it becomes difficult to evaluate the performance of data centers. Acquisition of data from multiple sources and filtering it out for consumable information was a challenge. Once the data was acquired, it was followed by analytical studies to find patterns that could then be used to solve part of the complete problem.

1.4 Project Objectives

To enable recognition and automatic remediation of problems raised by SDDCs provisioned to clients. The solution to the problem at hand has been divided into several smaller subproblems, that build into one single final product –

- Finding a way to collect data without causing a lot of stress on the hypervisors memory and network bandwidth.
- Organizing received metrics in a standardized structure that can then be read with ease from the database and analyzed effectively.
- Break the kernel into multiple layers and find how one layer affects the next one.

- Introducing false bugs and code delays and analyzing the received metrics to find the top most critical attributes that get affected by these manually induced errors.
- Build a model that can sense and report anomalous behavior in the critical attributes selected.
- Creating a software system that runs every few seconds to proactively analyze received data and present the analysis in a compact and effective form.
- Creating a graphical interface that will display the health of various components and report all services that will be affected by any misbehaving component.
- On finding an error and the root cause, automatically calling a script or a run book to mitigate the issue as soon as a drop in performance is noticed.

1.5 Target Specification

The software when in production will automate detection of errors and remediation for every single registered SDDC. It will do so by continuously monitoring all the registered SDDCs. In case it cannot remediate by itself, it will notify the reliability engineers with information about where the failure possibly originated from.

Chapter 2

Literature Survey

2.1 Virtualization Technology

Virtualization is the process of creating a software-based (or virtual) representation of something rather than a physical one. Virtualization can apply to applications, servers, storage, and networks and is the single most effective way to reduce IT expenses while boosting efficiency and agility for all size businesses.

A virtual machine is a software computer that, like a physical computer, runs an operating system and applications. The hypervisor serves as a platform for running virtual machines and allows for the consolidation of computing resources. Each virtual machine contains its own virtual, or software-based, hardware, including a virtual CPU, memory, hard disk, and network interface card. Because virtual machines are decoupled from specific underlying physical hardware, virtualization allows you to consolidate physical computing resources such as CPUs, memory, storage, and networking into pools of resources that can be dynamically and flexibly made available to virtual machines.

2.2 VMware Products

This section covers a brief introduction of the various products [1] being used during the project.

2.2.1 VMware ESXi

VMware ESXi is a type-1 hypervisor developed by VMware for deploying and serving virtual computers. As a type-1 hypervisor, ESXi is not a software application that is installed on an operating system; instead, it includes and integrates vital OS components, such as a kernel.

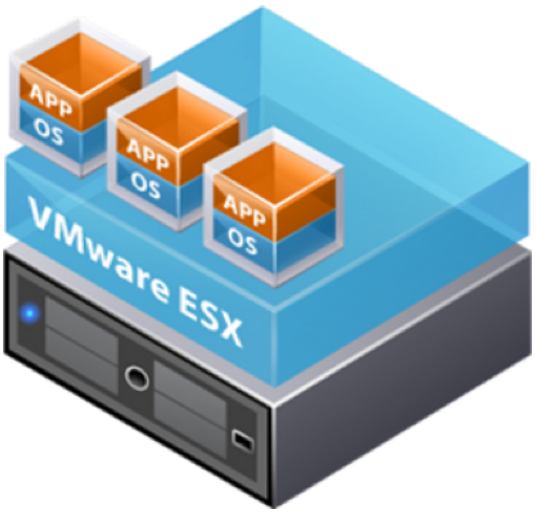

Figure 2.1: VMware ESXi

The VMware ESXi hypervisor is a lightweight hypervisor that runs over the volatile memory of the physical server. It can support running up to 64 virtual machines on a single node. ESXi provides a web interface through which a user can monitor system health and launch virtual machines with ease. The hypervisor is the single most important component of this project, since metrics are collected on a host level.

2.2.2 vCenter Server

VMware vCenter Server is a data center management server application developed by VMware Inc. to monitor virtualized environments. An admin can launch a single instance of vCenter Server Appliance (VCSA) and monitor multiple data centers from a single management console [2]. VCSAs can be installed to manage different sizes of data centers. For this project, the weakest configuration suffices as it can handle up to 10 ESX hosts.

The appliance also exposes a web API which can be used to manipulate all component machines and clusters with the help of scripts. Given in Figure 2.2 is the hierarchical structure of how VCSA operates.

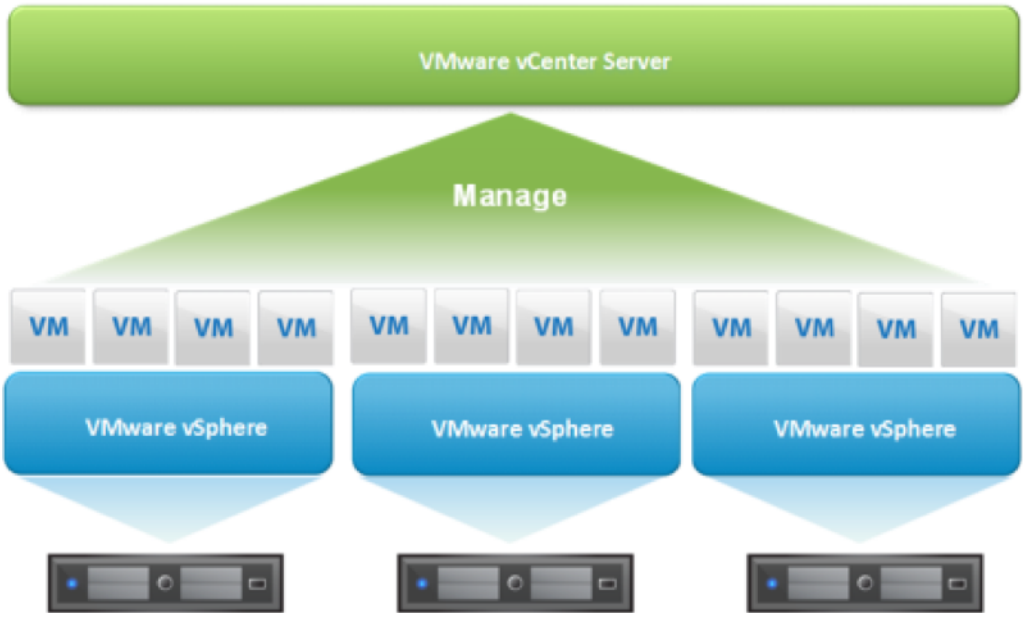

Figure 2.2: vCenter Server Appliance hierarchy

vCenter Server is installed at the primary server of a virtualized data center and operates as the virtualization or virtual machine manager for that environment. It also provides data center administrators and a central management console to manage all the system's virtual machines.

2.2.3 vSAN

VMware vSAN is a hyper-converged, software-defined storage (SDS) product developed by VMware that pools together direct-attached storage devices across a VMware vSphere cluster to create a distributed, shared data store. The user defines the storage requirements, such as performance and availability, for virtual machines on a VMware vSAN cluster and vSAN ensures that these policies are administered and maintained. vSAN is part of the VMware ESXi kernel.

A virtual SAN appliance allows unused storage capacity on virtual servers to be pooled and accessed by virtual servers as needed. This can be managed through the web client provided by the VCSA.

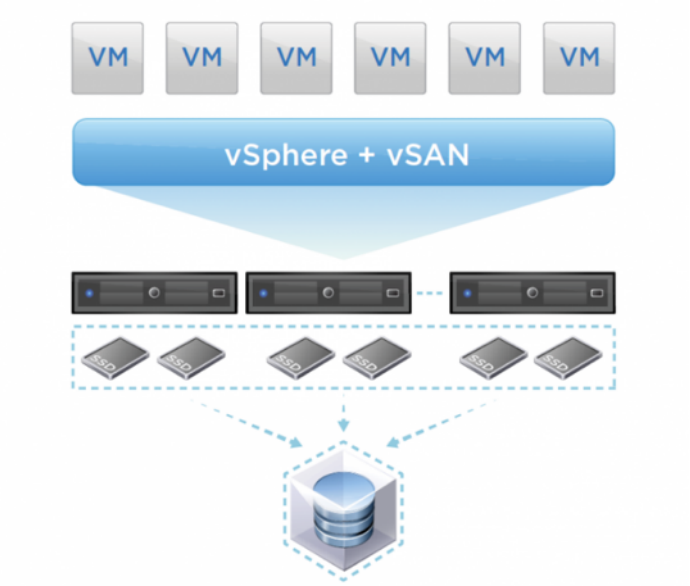

Figure 2.3: Hyperconverged vSAN system

vSAN supports both hybrid and all-flash configurations using a two-tier architecture. Both configurations use a caching tier and a capacity tier. The caching tier is composed of at least one flash device per host. The capacity tier is composed of at least one flash device (for all-flash) or one magnetic disk (for hybrid) per host. vSAN combines the host's storage resources into a single, high-performance, shared data store that all the hosts in the cluster can use.

2.2.4 vMotion

vMotion enables the live migration of running virtual machines from one physical server to another with zero downtime, continuous service availability, and complete transaction integrity. VMotion is a key enabling technology for creating the dynamic, automated, and self-optimizing data center.

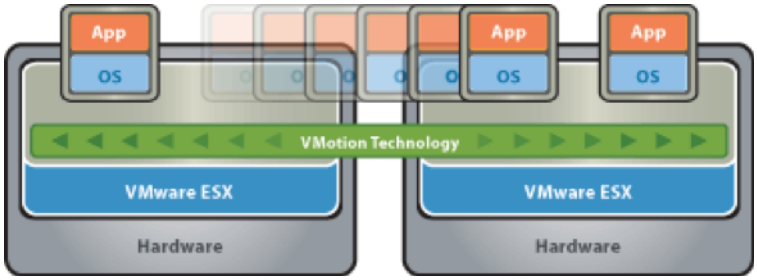

Figure 2.4: vMotion in action

2.3 Statistical Analysis and Machine Learning

Machine learning is the science of getting computers to act without being explicitly programmed. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or in-feasible.

Machine learning tasks are typically classified into two broad categories, as shown in Figure 2.5, depending on whether there is a learning signal or feedback available to a learning system –

- *Supervised learning* – The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

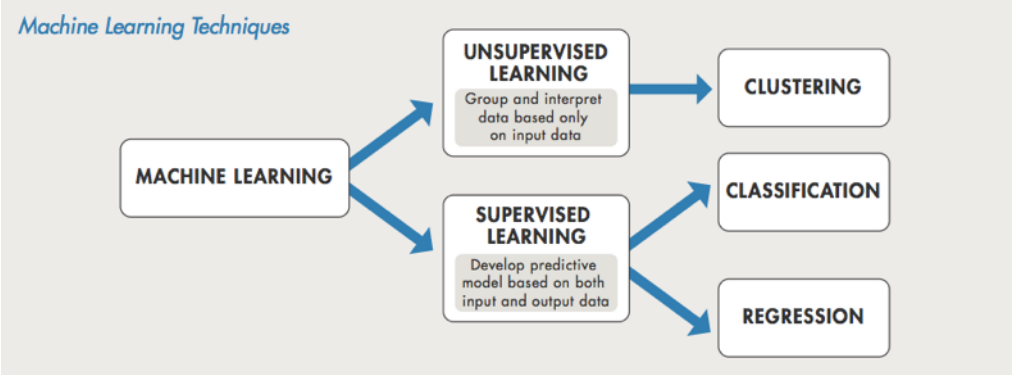

Figure 2.5: Types of machine learning

- *Unsupervised learning* – No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end.

Statistics is a branch of mathematics dealing with the collection, analysis, interpretation, presentation, and organization of data. Statistics is a precursor to any kind of machine learning, and all ML models rely on some form of statistical inference [3].

The next few sub-topics describe some essential statistical techniques and machine learning concepts that were part of this project.

2.3.1 Control Charts

Control charts are representations of the quality of a metric over a period of time. Metrics come in as continual time series data and this continuous data is used to determine a normal quality range. The quality range is said to be the normal operating range. Figure 2.6 is a typical control chart with its quality ranges upper bound and lower bound marked by UCL and LCL respectively [4].

The quality range is typically calculated using standard deviations and averages of historical data. Most data that occurs naturally tends to fit in a

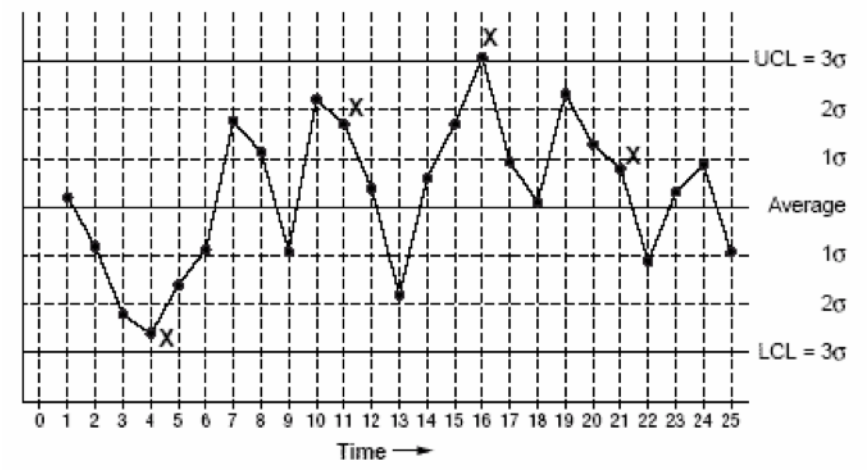

Figure 2.6: Control chart

bell-shaped curve a normal distribution of values. A very well-known result that follows up from normal distributions is that most of the data (99.7%) falls within a range of 6 standard deviations of the population [5]. A visual representation of normal distributions can be seen in Figure 2.7.

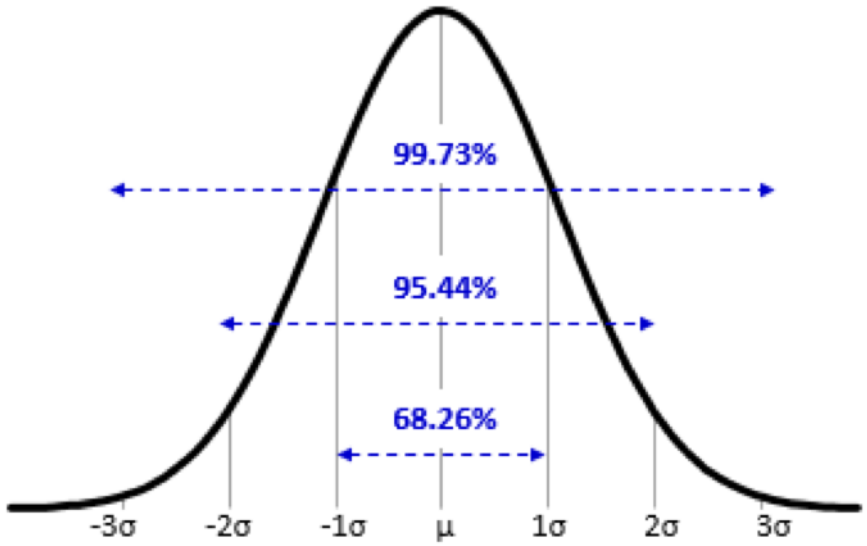

Figure 2.7: Normal distribution in data

In Figure 2.7 above, μ corresponds to the mean of the data and corresponds to a single standard deviation of the data points. Since most of the data is known to fit within 6 standard deviations, it is common to use the six-sigma rule to set the quality bounds for control charts. Hence, a normal upper

bound is the value +3 and similarly, the lower bound is determined by the value $\mu - 3\sigma$. In short, $\mu - 3\sigma <$ Acceptable Value $< \mu + 3\sigma$.

A lot of the data center monitoring techniques that are already in use rely on the method of control charts to determine if there is any fault in the metric values. As part of this project, control charts were implemented as a first step. However, while this approach was extremely simple to implement, there were a few drawbacks because of which more robust methods were required.

2.3.2 K-Means

K-means clustering is a type of unsupervised learning, which is used when you have un-labelled data (data without defined groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K, i.e., when K=2, K-Means divides the set of points into 2 groups. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. Figure 2.8 shows the result after applying clustering to a set of points with K = 2.

The K-means algorithm can use one of the many distance measures to determine the cluster centroids. The choice of distance measures depends on the type of data thats coming in, the distribution of data and the number of dimensions. In this project, two major types of distance measures (Euclidean distance and Manhattan distance) were used, and both gave similar results. However, since Manhattan distance is less computationally intensive to process, that was the preferred choice.

2.3.3 Kernel Density Estimation

Kernel Density Estimation is a technique to estimate the unknown probability distribution of a random variable, based on a sample of points taken from that

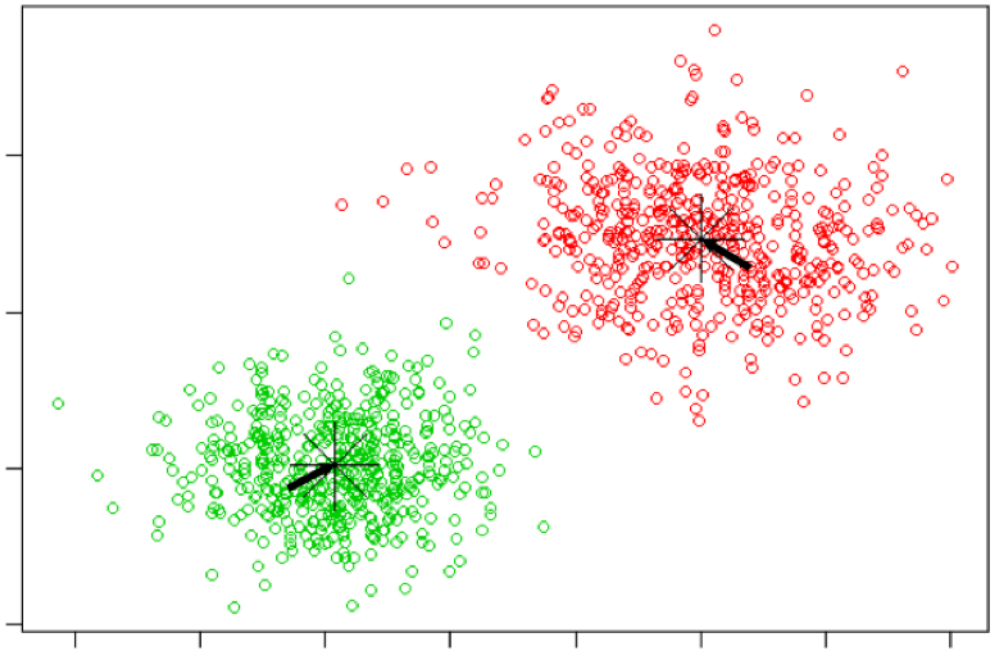

Figure 2.8: K-means in action

distribution. It is a non-parametric algorithm used to estimate the probability density function (PDF) of the variable, and we can use several kernels to do this, hence the name [6].

Figure 2.9 gives us a general idea of how the PDF is determined given the data points using one of the kernels. Each data point is assigned a strength (the kernel function), and in the end all the kernels over individual points are added to get an over kernel density estimate function. Data points that are closer together thus form higher density peaks. This density value can then be used to get a probability estimate of the point.

Generally, and in this project, a Gaussian (normal distribution) function is used as the kernel. Apart from the kernel itself, the KDE algorithm has a free parameter, known as the bandwidth. This bandwidth determines the spread of the kernel and is an important parameter that helps determine the overall shape of the probability density function so formed. Figure 2.10 given below shows us the importance of the bandwidth selected.

As can be seen, when the bandwidth is too low (0.05), the probability

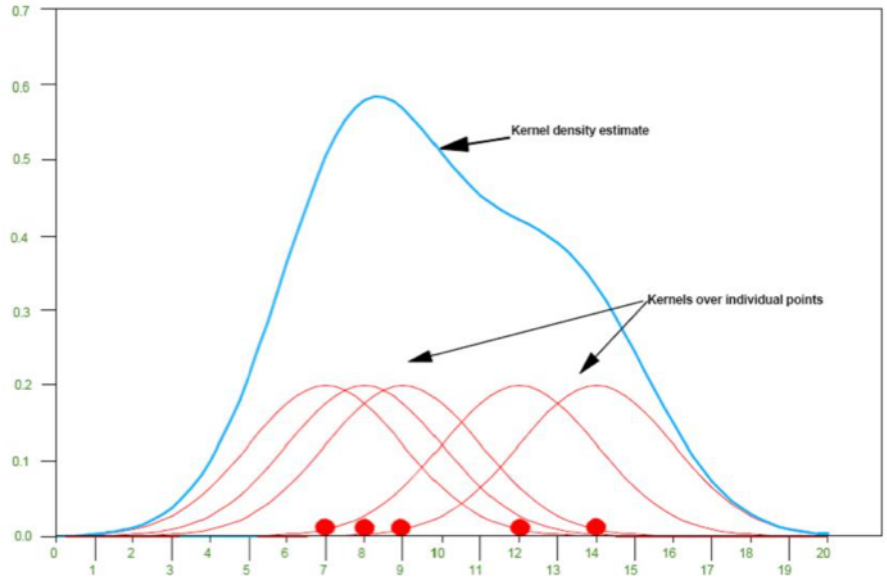

Figure 2.9: Kernel Density Estimation

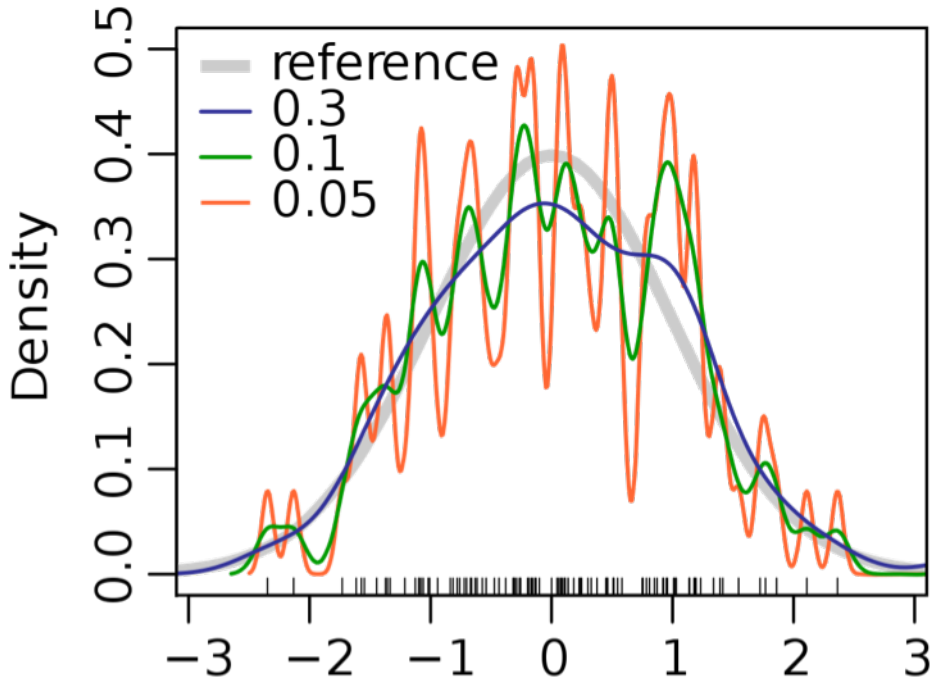

Figure 2.10: Bandwidth in KDE

density function is immensely distorted. The value of 0.3 suits the distribution almost perfectly. On the contrary, if the bandwidth value is too high, the density function received might be over-smoothed and may construe wrong results.

2.4 Introduction to the Project Title

Given the problem statement, this project mainly aims at solving the issue by maintaining an overall health of the system. It does so by monitoring various metrics that are part of the multiple functional transactions performed by systems. Monitoring can be of two main types –

2.4.1 Reactive Monitoring

This mode of monitoring relates to a push model. Once a transaction has taken place, it triggers the reactive monitoring agent to check if the transaction that took place was normal in comparison to other similar transactions. Data collection in reactive monitoring happens at inconsistent intervals and is solely triggered by an action.

2.4.2 Proactive Monitoring

On the other hand, proactive monitoring corresponds to a pull model. Unlike reactive monitoring, proactive monitoring collects metrics at a fixed interval of time, regardless if there is an action being performed or not. In other words, it does not depend on the state of the system. While proactive monitoring is performed at fixed intervals, the values of metrics may not be as consistent as those of reactive monitoring.

Reactive monitoring triggers monitoring checks after an action has been completed. In our use case, there is a need of proactive monitoring since we would like to have information about a systems health at any given point

of time. Since we would like to know about a systems failures as soon as possible and preferably before it impacts any user-initiated actions, proactive monitoring fits our requirements perfectly well.

An amalgamation of the above described terms gives rise to the projects title *A Proactive Health Monitoring Tool for SoftwareDefined Datacenters*.

Chapter 3

Methodology

The first part of this chapter talks about the architecture of the software. This is subsequently followed by detailed implementation level information of every single component present in the architecture.

3.1 Software Architecture

Any given software is a collection of smaller modules that serve their own purpose. This project is no different. The software can be broken down into smaller functional units that when put together achieve the desired objectives. An overall structure of the project is given below in Figure 3.1.

The software is implemented on a layered basis with the lowest layers corresponding to the physical devices and kernel functions and the upper layers corresponding to services and products that depend on these lower layers. Identifying these separate layers are subject to experimentation and following the kernel code.

As an example of layers, consider the activity of live migration (vMotion). For a vMotion to happen, the pages in memory are repeatedly written at the destination through the network multiple times during the precopy phase. Once this is completed, the storage file system is migrated by reading data sectors into the memory and transferring them over the network. Hence, an

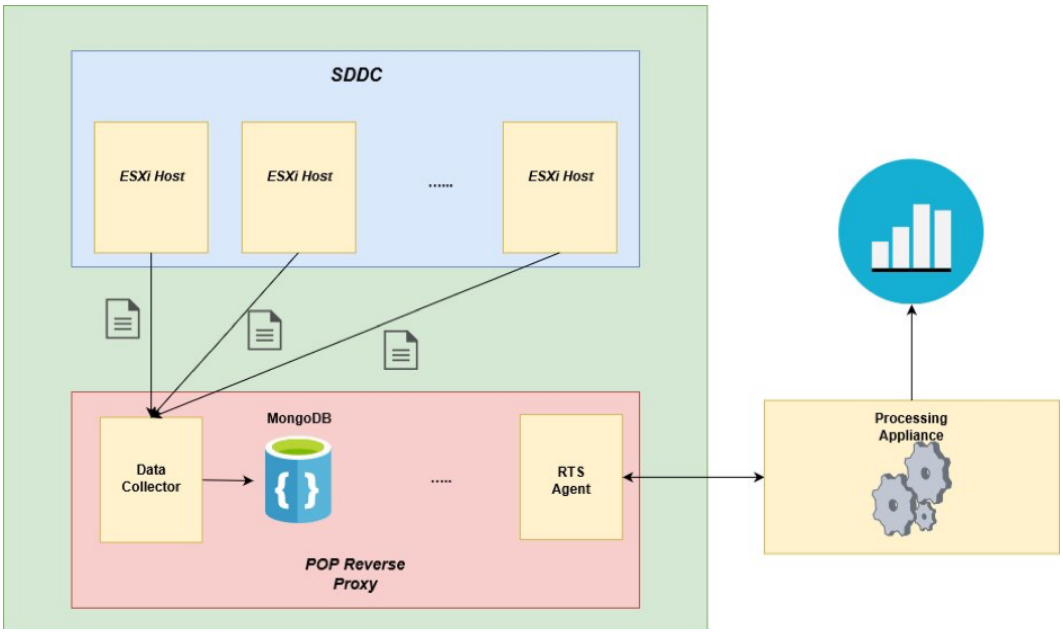

Figure 3.1: Software Architecture

error at the network layer will slow down all the other processes involved in the live migration process, making it the lowermost layer. Any other critical component is added in the lower layers. For the vMotion service, some layers are mentioned in a decreasing level of impact (higher layer) in Figure 3.2.

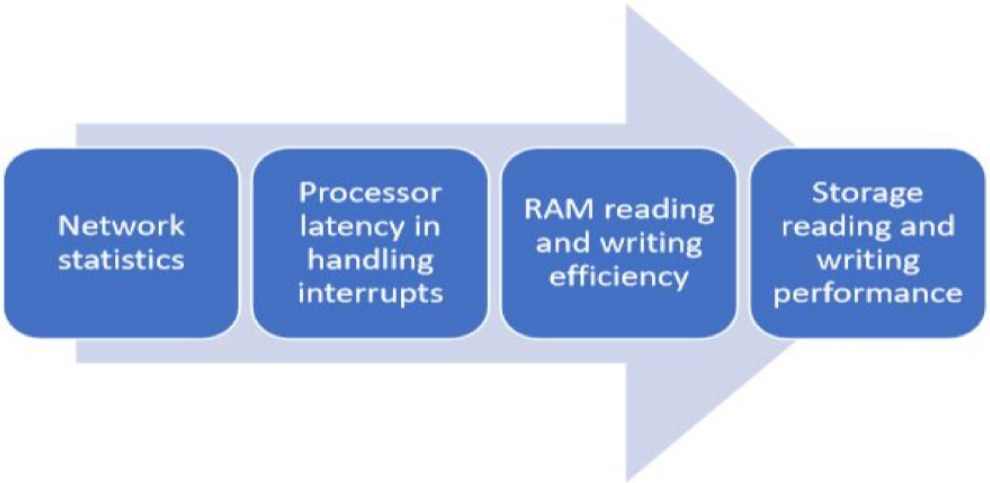

Figure 3.2: Layers evaluated from high impact to low impact components

As a whole, the software consists of 4 main components the data collector, data processor, the UI and the Remediation and Troubleshooting System

(RTS). The whole software can be summarized by Figure 3.3.

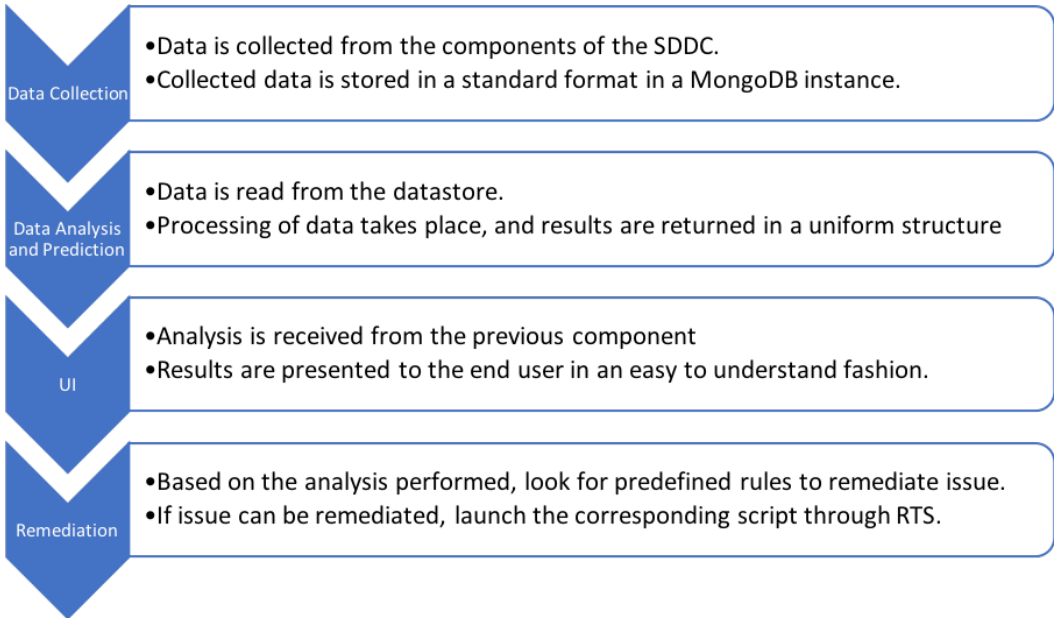

Figure 3.3: Software Workflow

3.1.1 Data Collector

Each SDDC contains various software defined hosts. The health of the SDDC itself depends on the health of the hosts that constitute it. These hosts are continuously polled by an external system for their metrics [7]. Since the hosts run ESXi hypervisors on them, it is impossible to run any new software on them. The developers want the ESXi to have as less bloat as possible, since the hypervisor runs on RAM and not on permanent storage.

Data collection can hence only be carried out using the tools that are already built into the hypervisor. Some examples of these tools are

- **VMKernel Sys Info Shell (Vsish)** This is an extremely important tool that keeps track of kernel level logs and can be easily read by a python library called vsi. Vsish is built into all ESXi versions and gives us direct information related to hardware devices and services. This information is stored into nodes and is updated at a very high frequency.

- **Tcpdump-uw** A barebones implementation of the tcpdump command normally found on Linux distributions. For any network related information, a tcpdump is taken for a given set duration and its characteristics are analyzed. Some features like the number of duplicate packets and the round-trip time give us accurate information about the health of the network.

Along with normal data, collection of erroneous data is also required. The VSI shell has a few nodes that can have their values manipulated to emulate hardware faults. For example, there exists a node that enables us to set a delay at the storage adapter level, for each IO that takes place between the datastore and any connected virtual machine.

3.1.2 The Database

MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. The fact that its a NoSQL database gives us the flexibility to readily modify the fields of data we need to record, unlike SQL databases which must define a schema before storing any data. Even though MongoDB was utilized for the project, the code is flexible enough to accommodate any change in the database.

3.1.3 POP Reverse Proxy

In computer networks, a reverse proxy is a type of proxy server that retrieves resources on behalf of a client from one or more servers. These resources are then returned to the client as if they originated from the Web server itself. Figure 3.4 shows us the position and the importance of a reverse proxy.

Since all companies value their privacy, VMware has made sure that any given host on the SDDC or the vCenter Server Appliance is not directly accessi-

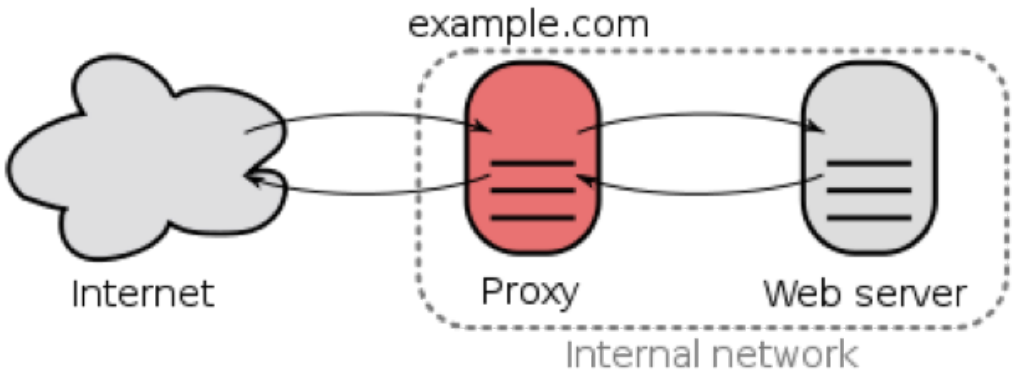

Figure 3.4: Reverse Proxy

ble from the internet. Instead, all requests for connection must be mandatorily routed through a reverse proxy (internally known as POP). Since POP is the only single point of entry to an SDDCs internal network, our processing appliance can only read data through one of the many agents running on it. All the data collection code must also be run on the POP since the processing appliance cannot directly contact the ESXi hosts.

3.1.4 Processing Appliance

The data processing component runs every x seconds (configurable frequency). It reads the last 2000 records written to the database and processes it to find any anomalies detected in the system. This program contacts the POP Reverse Proxy for all the SDDCs to be monitored and fetches the data to be processed. Starting from the lowest layer, the software moves to the higher layers and reports any anomaly it finds in the process.

The process of anomaly detection happens through a novelty detection algorithm that is explained in detail in the implementation details. Once an anomaly has been detected, the application takes the necessary steps by either raising an alert on a dashboard or by taking an action through the Remediation and Troubleshooting System.

3.2 Data Analysis and Error Prediction

Perhaps the biggest challenge for any monitoring project is picking the right amount of data and the right metrics. Both too little data and too much of noisy data can harm analysis and prediction. At first, a control chart technique (Explained in 2.3.1) was used to determine normal upper and lower bounds for data. However, due to reasons stated above, the control chart technique was not always accurate.

In fact, not only the control chart technique, but most of the error detection techniques were extremely susceptible to erroneous values. So, a new novelty detection algorithm was researched upon which was not sensitive to erroneous values. This new error detection algorithm also gave us a better upper thresholds for values in the case where the standard deviation of the values was close to, if not zero. Data that comprised of a lot of similar values also posed a problem to the control chart technique since it would result in a very low standard deviation which in turn would result in an acceptable range that was not very tolerant of incoming values.

The algorithm that was finally used comprised mainly of two parts Clustering and Probability Estimation to find an upper threshold.

3.2.1 Clustering

Since all the novelty detection algorithms were heavily affected by erroneous values, there was a need to get rid of values that were significantly far away from the normal trend followed by the data points. To deal with this, a new approach of K-Means, Repetitive K-Means was used as a pre-processing step. Figure 3.5 is shows the flow of this pre-processing algorithm, and is followed by an explanation.

The algorithm goes through multiple steps before returning with the filtered data, as explained below

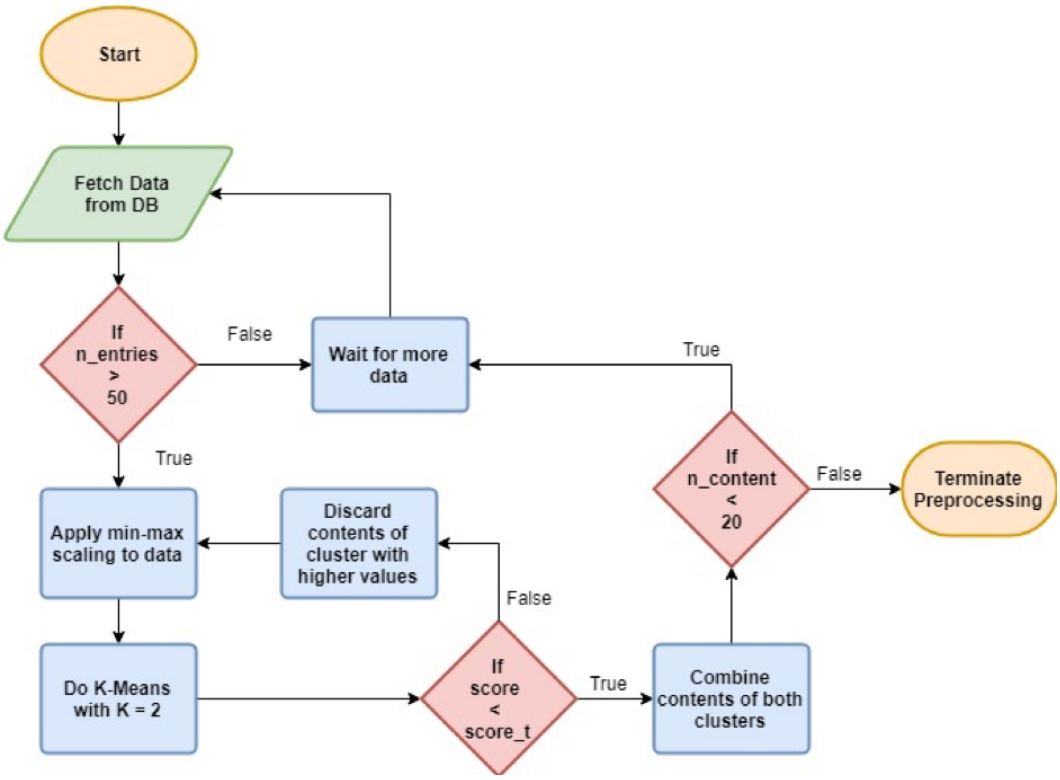

Figure 3.5: Repetitive K-Means algorithm

- Time Series data is read from the data source (POP Reverse Proxy in our case) on a per-host, per-layer basis. If sufficient non-zero records exist on the database, the data processing starts, otherwise we wait for more data to be written into the database from the data collector.

- Once we have enough data, we apply a min-max scaling over the values over the range 0 to 1, i.e., the highest value present will now be scaled down to 1, and the lowest value will be scaled down to 0.

- Then we apply K-Means and form 2 clusters out of the given data.

- The distance between the two centroids so formed will be in the range 0 to 1. This distance is denoted by score. score_t is an adjustable hyper-parameter which gives us a minimum score for termination, i.e., if the value of score is greater than the threshold score, score_t, it implies that the 2 clusters formed are still farther from each other than we would like.

- If the clusters are far enough, we pick the cluster with smaller values (since we would always ideally want latencies to be low), and repeat from the min-max scaling step until the score is lesser than the threshold set.
- Once this condition is satisfied, we check if we are left with enough clean data points. If not, we wait for more data to come in and then try again.
- When we have enough data points (hardcoded to 20 points in the project), we can feed these valid data points in to the next Kernel Density Estimation step.

3.2.2 Kernel Density Estimation

Once we have collected enough clean data, we move on to forming an estimation model which can then be used to set upper thresholds on latencies. Unlike the pre-processing clustering step, Kernel Density Estimation must happen much more often to make sure that the thresholds remain updated depending on the most recent values and never get stale.

For the project, we have taken the a Gaussian (normal distribution) kernel and a bandwidth of 1. The thresholds are set by finding the point at which the area under the probability density function (PDF) sums to 0.995. A flow of the second part of the data analysis is show in Figure 3.6, followed by an explanation.

The steps involved in the algorithm are as follows –

- Start by pre-processing the data using the repetitive K-Means clustering explained in the previous topic.
- Form a KDE model using this clean data to ensure that the probability density function obtained is uniform.
- Determine the upper threshold by finding the point at which the area under the PDF reaches 0.995 (configurable value). This can be done by

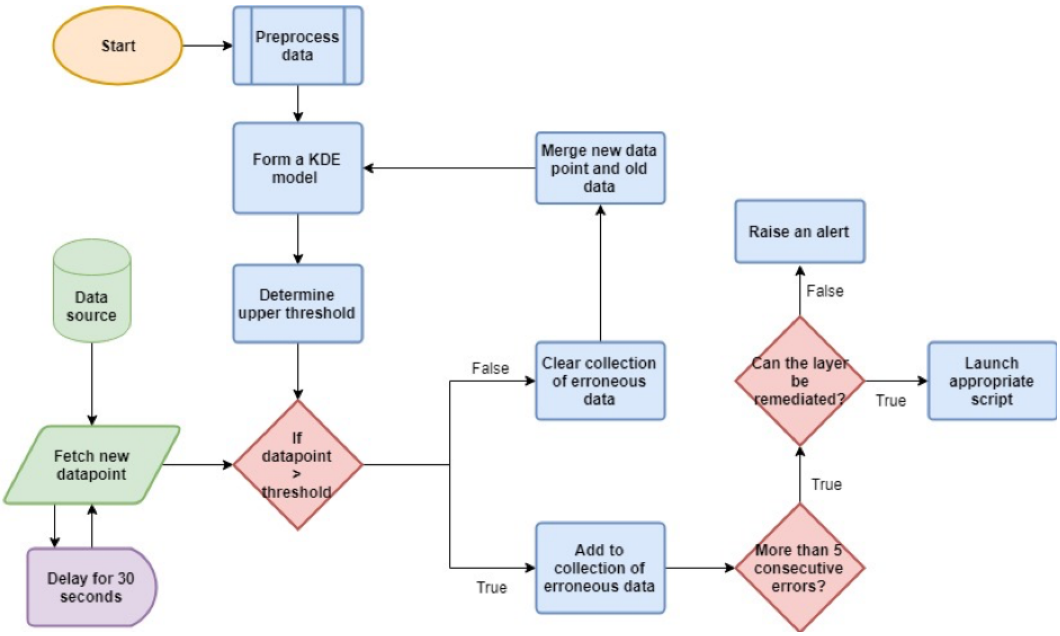

Figure 3.6: Kernel Density Estimation Flow

iteratively checking for an integral point where the cumulative density function just crosses the required value.

- This upper threshold can then be used to compare new data points. If the incoming data points are larger than this threshold value, we can keep track of them in a separate collection.

- If the last few values have been consecutively above the normal threshold set by KDE, there is a certain problem that needs to be addressed. If a rule exists to fix the said problem, apply it. Else, raise an alert so that someone could look into it.

- If a normal value comes in, we queue it so that it can be included in the KDE when the next threshold is to be calculated. This ensures that the threshold is ever-changing and accounts for slight changes in trends.

- The KDE model is updated every x iterations (x = 10 for this project), where this x could depend on how frequently the user wants to change the thresholds. A very high frequency can lead to slowdowns since computing

KDE is computationally costly.

3.3 Remediation and Troubleshooting Service

The product is composed of the following three components -

- *RTS UI* – It renders an interface developed using Angular 2. It is a single central server that has access to all SDDC instances. It gives the ability to view or execute runbooks. It also gives the ability to execute individual scripts and lets user see the progress of the script execution. The UI will show task execution progress, results, errors, and logs.
- *RTS Server* – It provides the infrastructure to connect to the software defined datacenters and perform operations on them. The features of RTS server are listed below -
  - Library of Runbooks. A runbook is made up of several automation scripts (Modularity is implemented in the automation scripts that is they can be added or removed without effecting others)
  - Provides REST APIs for each script.
  - These scripts can call RTS Agent and the other SDDC components (VC, PSC, NSX, ESX)
  - It suupports asynchronous and synchronous script execution mechanisms.
  - Script execution progress and state can be monitored.
  - Keeps the list of previous script executions together with invocation parameters, performance metrics, execution results, errors, logs.
  - Provides direct management capabilities for services running within the ESXi hosts, VCenter appliance and NSX cluster nodes. The

capabilities include starting, stopping, restarting and checking the status of these services.

  - It can also generate log bundles for troubleshooting the components.

- *RTS Agent* – The Remediation and Troubleshooting Agent is an intermediary web service inside the SDDC PoP. It provides the framework to execute commands on SDDC components like VC, ESX, NSX, and VSAN in a controlled and audit-able manner. It reduces the need for SSH access to a SDDC components and protects the stability of these components. The commands are exposed to the users as REST APIs. Command line interface to access the APIs is also available. The agent has a minimal footprint and is stateless. The features of RTS Agent are listed below -

  - Modular APIs to promote ease of use and maintainability. API additions or removals do not impact the functionality of other APIs within RTS Agent.
  - Ability to access any SDDC component (VC, PSC, NSX, ESX)
  - It uses SSH, SCP, etc. to access SDDC components
  - Access to APIs need to be protected by an authentication mechanism.
  - Generates Logs for RTS Agent troubleshooting, and command audits

3.3.1 RTS Design

The architecture is shown in Figure 3.7. The workflow is outlined below –

1. Monitoring components collects monitoring data.
2. Monitoring data is pushed into Operational Insight (OI).

3. Rules and Alerts engine (Nagios) listens to OI and triggers an alert. The alert description contains http link to a Runbook together with context (sddc_id, esx_id, ...).

4. User clicks the Runbook http link. Context is passed as parameter.

   (a) RTS Service Runbook manager shows Runbook document in browser UI.

   (b) Runbook document contains Execute buttons for automated steps.

5. User clicks the Execute button. Context is passed as parameter together with id of script to execute.

   (a) RTS Service Script manager starts an async task in SS Base to execute the script. Script makes one or more calls to SDDC components and RTS agent.

   (b) UI tracks task execution and provide updates (spinner or percentage).

   (c) Task completes. UI shows task status as success or failure. Failed tasks provide a link to failure details (error message or exception text).

3.3.2 Remediation procedure through RTS

The software developed for anomaly detection identifies the layer where the problem occured. To remediate the issue, commands can be executed on the SDDC through RTS.

The general tools available within RTS that can be used for solving these problems are-

- **Log bundle collection** - All the products including vCenter, NSX, ESX etc save the logs in the file-system of their hosts at a specific location.

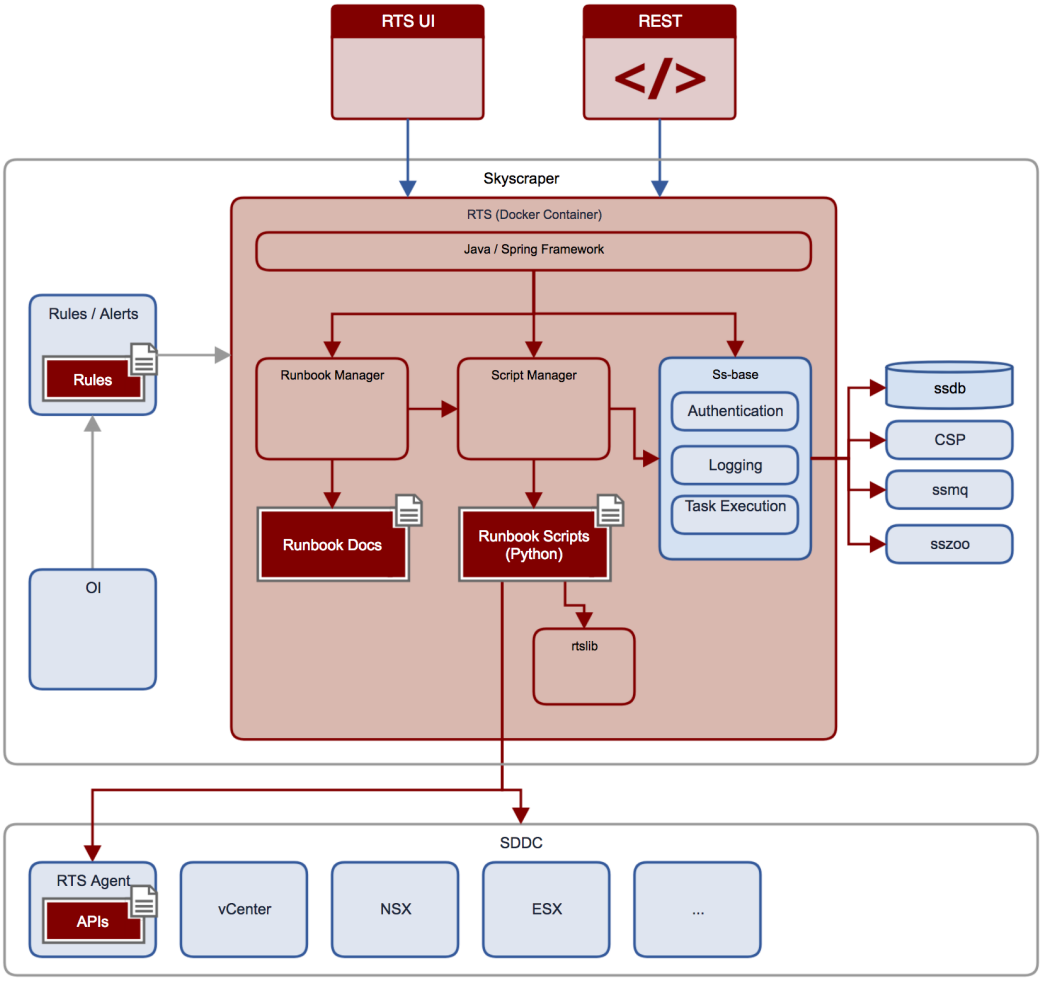

Figure 3.7: RTS Architecture

These bundles can be downloaded to the RTS server and then onto our local machines using certain scripts and then can be used for debugging purposes. The Log collection UI is shown in Figure 3.8.

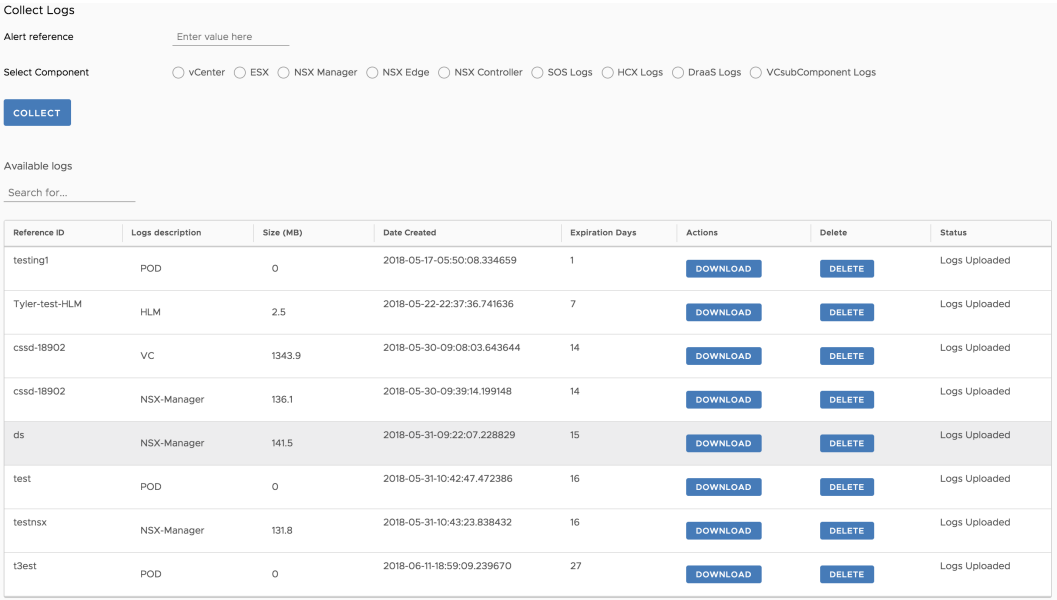

Figure 3.8: Log Collection Interface

- **Managing the services** - The system level processes/ services (usually, *Systemd* services) running in the appliances which make up the components work can be started, stopped or restarted using RTS after checking their present status.

- **Runbooks** - Runbooks constitute a pre-defined series of scripts which are ran together in a sequential manner to solve a specific problem. They are created when a problem is solved using a series of scripts. That sequence is then recorded and stored so that it can be used for solving a same or similar problem in the future.

- **Alarms** - Most products are configured to raise alarms when a system metric is out of place (for example, the network latency is high or the storage limit is about to be exceeded). Alarms from all these products are aggregated and listed in RTS and hence, can be used for troubleshooting.

3.3.2.1 Example scenario

An example of the layer where the problem was detected and the corresponding scripts/runbooks that can be used for remediation is shown below-

- **Problem detected in layer:** *vsan*
- **Scripts that can be used:** The scripts in Figure 3.9 correspond to the automation scripts related to vSAN. They range from checking the status of the services, monitoring the performance object's space usage to seeing the results of the last health tests. A sample execution ouput for the script which checks the liveliness of the clomd service is shown in Figure 3.10.

Script ID	Summary	Component
vsan_test_results	Get the results of a vSAN Health test.	vSAN
vsan_pref_service_manage	Enable the performance service	vSAN
vsan_physical_disk_overall_health	Get the result of the physical disk overall health check	vSAN
vsan_overall_health	Get the overall vSAN health status	vSAN
vsan_clomd_liveness	Get the status of clomd	vSAN
vsan_performance_object_space_usage	object space consumption	vSAN

Figure 3.9: vSan Scripts

- **Runbooks that can be used:** The runbooks shown in Figure 3.11 correspond to the runbooks related to vSAN. They include runbooks related to checking the status of the *Stats database*, checking the overall status of the physical hard disks etc.

- An expanded runbook which is used to check the overall health of the physical disk is shown in Figure 3.12.

Figure 3.10: Sample execution output

Runbook ID	Summary	Component
sddc_vsan_physical_disk_health_retrieval	Check physical disk health retrieval	vSAN
sddc_vsan_stats_db_check	Check vSAN Performance object space usage	vSAN
sddc_vsan_clomd_liveness_error	Fix errors with vSAN clomd liveness	vSAN
sddc_vsan_physical_disk_overall_health	Check overall physical disk health.	vSAN

Figure 3.11: vSan Runbooks

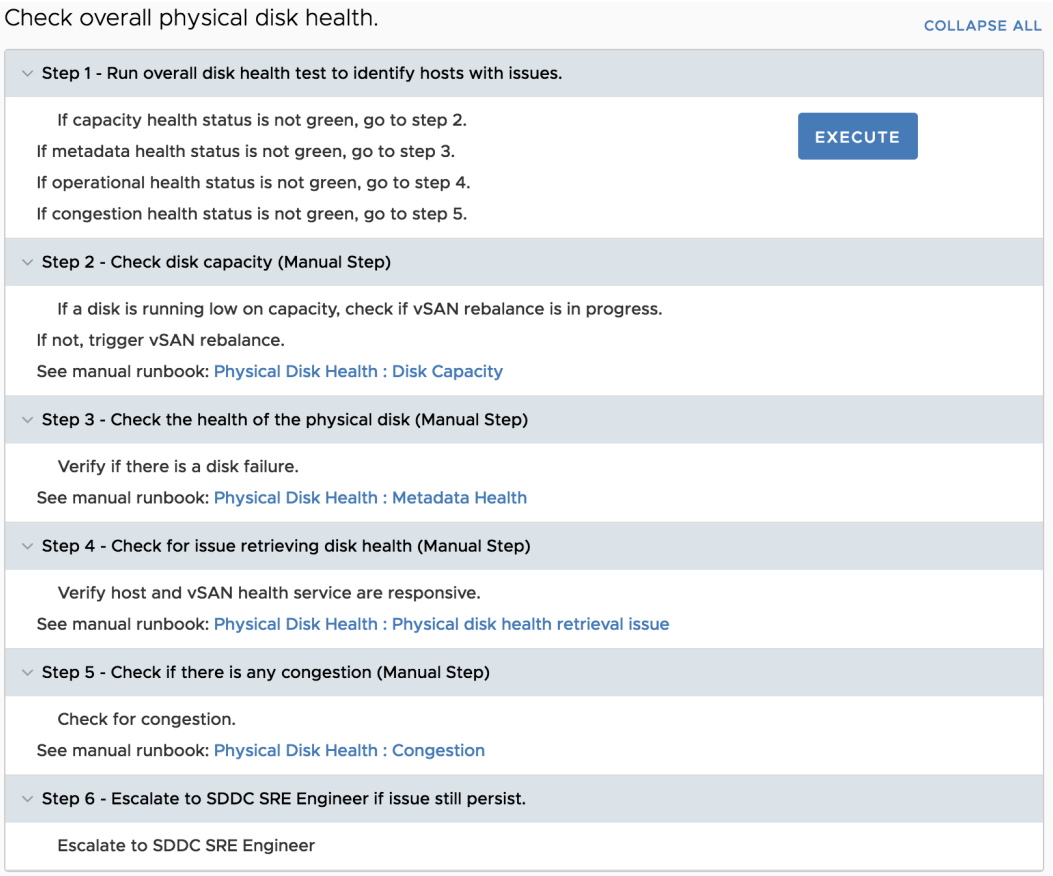

Figure 3.12: A runbook to check the overall physical disk health

Chapter 4

Results

VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware delivering a highly scalable, secure and innovative service that allows organizations to seamlessly migrate and extend their on-premises VMware vSphere-based environments including vSphere, vSAN, NSX, and vCenter Server to the AWS Cloud running on next-generation Amazon Elastic Compute Cloud (Amazon EC2) bare metal infrastructure. VMware Cloud on AWS brings the broad, diverse and rich innovations of AWS services natively to the enterprise applications running on VMware's compute, storage and network virtualization platforms. The service automatically provisions infrastructure and provides full VM compatibility and workload portability between your on-premises environments and the AWS Cloud [8].

This collaborative venture has proven to be profitable to both the companies with the customer base rapidly increasing. Reliability and availability of the product thus poses an important challenge. Our software helps in solving the problem in the following ways –

- It provides a common interface for data collection from several largely unrelated sources which may be relevant to the problem. Right now, VMkernel Sys Info Shell, TCP dump, VProbes and some REST API endpoints provide the required data but this list can be easily extended

as and when required.

- It provides a common algorithm using K-Means and Kernel density estimation which can be used throughout the stack of layers to recognize problems in the infrastructure by analyzing different metrics from each of the layers.

- The algorithms are self-adjusting and self-improving. They align with the general trend of the metrics during a particular period of time to decrease the detection of false positives.

- The software is flexible enough to introduce more layers and parameters to monitor pretty easily as and when the architecture of the SDDC components change.

- The layer in which the problem is detected is directly obtained from the software. Corresponding scripts and runbooks pre-existing in RTS can be used to remediate the issue. These automation scripts have to be created once and then can be used forever to solve the same or similar problems that may be encountered later removing the need for manual interventions.

VMware's SRE team was created when it's VMC product was launched. Site Reliability Engineering (SRE) is a discipline that incorporates aspects of software engineering and applies that to IT operations problems. The main goals are to create ultra-scalable and highly reliable software systems. This software is a step towards achieving that goal.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

A complete product has been developed which will help in reducing the time taken by developers and engineers to root cause an issue and then resolving it in an industry level SDDC stack. It will also help in recording and comparing performance changes as well as removing bugs for the upcoming releases of the products without having to spend time on checking each product and each operation manually. The framework has been released internally and is being used by teams in Cloud Platform Business Unit. There are plans of making the product a fling so that people can provide feedback and suggestions based on which it can be launched as a standalone VMware product if the project receives positive responses during the fling period.

5.2 Future Work

There's a scope for improvement of this software and some of the following suggestions can be worked upon in the future.

- The layered dependency of all the products and components of the SDDC stack haven't been fully discovered yet. As these dependencies become

more apparent, additional metric parameters and additional layers can be added to the software. More data sources can be added as well.

- Presently, the UI which shows the status of the layers in a graphical format is a part of the software package itself. Later, the backend and the user interface can be decoupled. REST endpoints can be created at the backend side to query the state of the SDDC by the UI. The UI service will then construct and display the dependency graph with the states in a browser. This decoupling of micro-services can prove to be beneficial when there is a single dedicated backend server and many consumer clients. It will help in easy debugging and show significant performance improvements.

- RTS is a go-to tool for the entire SRE team for troubleshooting the SDDC issues. If this software can be integrated with RTS and can be accessed through it's UI itself, productivity of the engineers will be increased significantly as the switching time will be removed.

References

[1] Vmware products. [Online]. Available: https://www.vmware.com/in/products.html

[2] vsphere documentation. [Online]. Available: https://pubs.vmware.com/vsphere-51/

[3] Introduction to statistics. [Online]. Available: https://en.wikipedia.org/wiki/Statistics

[4] Control chart. [Online]. Available: http://asq.org/learn-about-quality/data-collection-analysis-tools/overview/control-chart.html

[5] 68–95–99.7 rule. [Online]. Available: https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule

[6] Kernel density estimation. [Online]. Available: https://en.wikipedia.org/wiki/Kernel_density_estimation

[7] Introduction to vmware software-defined data center. [Online]. Available: https://code.vmware.com/sddc-getting-started

[8] Vmware cloud on aws. [Online]. Available: https://aws.amazon.com/vmware/

Project Details

Student Details

Student Name	Nikhil Gupta		
Registration Number	140911466	Section/Roll No.	B/64
Email Address	nanonikhil@gmail.com	Phone No.(M)	7390932789

Project Details

Project Title	A Proactive Health Monitoring Tool for Software Defined Datacenters		
Project Duration	6 Months	Date of Reporting	10-01-2018

Organization Details

Organization Name	VMware Software India Pvt. Ltd.
Full Postal Address	Kalyani Vista, JP Nagar, Bengaluru(560076), India
Website Address	www.vmware.com

Supervisor Details

Supervisor Name	Priya Pandian		
Designation	Senior Manager		
Address	JP Nagar, Bengaluru(560076), India		
Email Address	kpandian@vmware.com	Phone No.(M)	9886172287

Internal Guide Details

Faculty Name	Raghvendra Achar
Address	Dept. of I&CT, MIT, Manipal(576104), India
Email Address	raghavendra.a@manipal.edu