Predictive Analysis and Development of Dimensioning Tool

Thesis submitted in partial fulfillment of the
Requirements for the degree of

Master of Engineering
ME (Big Data and Data Analytics)

By

Uday T

(181046016)

Internship start date: 03/06/2019

Internship end date: 30/04/2020

Under the Guidance of

External Guide:
Mr. Ravi Rajaram
R&D Specification Specialist
Nokia Networks
Bengaluru

Internal Guide:
Mr. Arockiaraj S
Assistant Professor
MSOIS, MAHE
Manipal

MANIPAL
ACADEMY of HIGHER EDUCATION
(Deemed to be University under Section 3 of the UGC Act, 1956)

MANIPAL SCHOOL OF INFORMATION SCIENCES

(A Constituent unit of MAHE, Manipal)

MANIPAL
ACADEMY of HIGHER EDUCATION
(Deemed to be University under Section 3 of the UGC Act, 1956)

MANIPAL SCHOOL OF INFORMATION SCIENCES

(A Constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that this thesis work titled

Predictive Analysis and Development of Dimensioning Tool

Is a bonafide record of the work done by

Uday T

(181046016)

In partial fulfillment of the requirements for the award of the degree of **Master of Engineering - ME (Big Data & Data Analytics)** under MAHE, Manipal and the same has not been submitted elsewhere for the award of any other degree. The dissertation does not contain any part / chapter plagiarized from other sources.

External Guide:
Mr. Ravi Rajaram
R&D Specification Specialist
Nokia Networks
Bengaluru

Internal Guide:
Mr. Arockiaraj S
Assistant Professor
MSOIS, MAHE
Manipal

Prof. Ramashesha C S
Director,
Manipal School of Information Sciences
MAHE, Manipal

ACKNOWLEDGEMENT

I am overwhelmed in all humbleness and gratefulness to acknowledge all those who have helped me to put these ideas, well above the level of simplicity into something concrete. To start off, I thank our dignified Director, Prof. Ramashesha C S, and the rest of the faculty at SOIS, Manipal, for all their help during my time in college. I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. Next, I would like to thank the System Performance team of NetAct at Nokia Networks, Bengaluru, with special thanks to my mentor, Mr. Ravi Rajaram for giving me the opportunity to work on real life problems and encouraging me in my journey of trying to solve them. I also express my gratitude to my final year project guides in college, Mr. Arockiaraj S and Mrs. Ravikala Kamath, whose contribution in stimulating suggestions and encouragement, helped me to go about my project. Special thanks go to my colleagues, who helped me along the way. I also appreciate the guidance given by other faculty members, especially in my project presentation that has improved my presentation skills thanks to their comments and feedback. Finally, I would like to thank my family, for all their support, and The Almighty for His blessings. I take pleasure in presenting before you, my project, which is the result of a blend of both research and knowledge.

INDEX

LIST OF FIGURES

CHAPTER 1: ABSTRACT

In this era where data and voice services are available at a push of a button, service providers have virtually limitless options for reaching their customers with value-added services. The changes in services and underlying networks that this always-on culture creates make it essential for service providers to understand the evolving business logic.

There are various procedures that must be taken care of in order to provide a seamless fault-tolerant and highly available Network Management System (NMS) and Element Management System (EMS).

In telecom networks the NOKIA NetAct continues to evolve with the new technologies and services in mobile broadband with larger data of gigabits and terabits. Network Management is used to recognize the best practices to improve the operational efficiency, communication in the operation teams, reducing the operational risks and it ensures operational resilience.

Network management features support the life cycle of networking technologies by testing some practices in simulation environment before deploying to the real environment. In these test scenarios with various loads and configurations huge data is generated which can be analyzed to find trends, patterns and deploy machine learning models.

CHAPTER 2: INTRODUCTION

This section gives the state of art to the past and present evolutions from which the technology has evolved into the system under consideration for this dissertation work. Need in simplicity in the field of mobile broadband technologies in telecommunication services and consistent network growth are driving towards the new direction in Operational Support System (OSS).

Several researches have been carried out in the field of Telecommunications. In telecom networks the NetAct continues to evolve with the new technologies and services in mobile broadband with larger data of gigabits and terabits. Network Management is used to recognize the best practices to improve the operational efficiency, communication in the operation teams, reducing the operational risks and it ensures operational resilience.

2.1 NetAct

NetAct consists of many tools for handling a number of network elements and expanding networks. It is designed for handling and increase in both complexity of the network and the amount of traffic and data. With NetAct both the network management and the services within the network are managed centrally enabling the user to view failures of the network element, quality indicators and traffic in a single screen view.

To facilitate its integration in multi-vendor environments, NetAct provides several open northbound interfaces (standard-compliant and proprietary ones) that offer to operators an unified management capability for the entire network. For this purpose, a mediation application function provided by a so-called Northbound Interface Agent performs the mapping between the management information received by NetAct or generated in NetAct and the information required by the upper level management systems from Network.

NetAct provides the following management functions as depicted in the Figure 2.1:

- Fault management
- Optimization
- Configuration management

- Performance management
- Security management
- Network administration
- NetAct administration

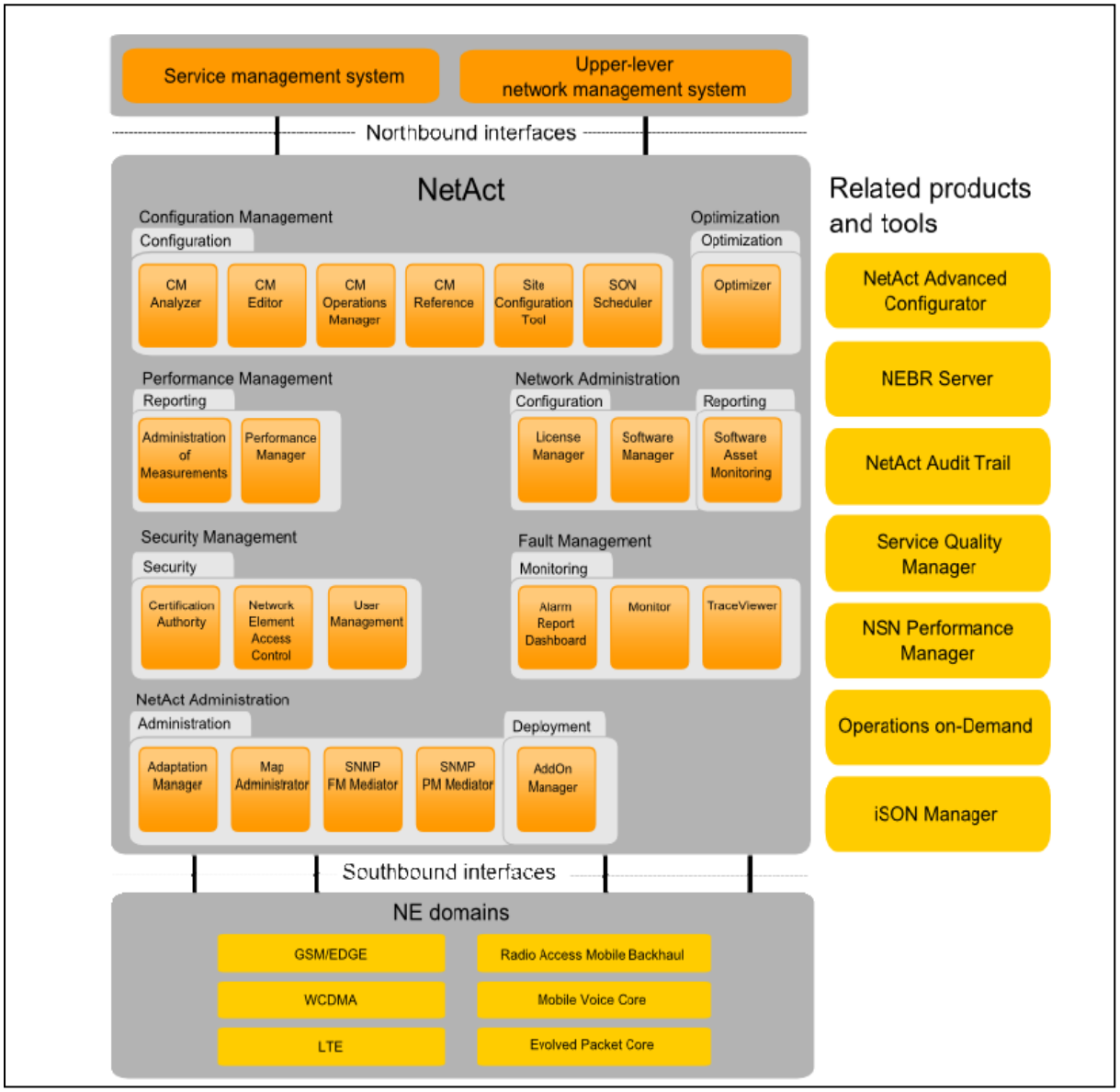

Figure 2.1 Overview of NetAct

In addition to NetAct applications there are also standalone tools that are dedicated to specific management functions and provide advanced management features, for example, NetAct Audit Trail for security management, and NetAct Advanced Configurator for configuration management. These standalone tools need a dedicated hardware platform and are sold separately. NetAct provides 24*7 services and monitors the live network and monitors different network elements by providing different applications like configuration management editor, performance manager, fault manager, network element manager and user management.

2.1.1 Definitions

NetAct provides the following management functions.

(a) Fault management

In NetAct, fault management enables user to manage alarms from various network elements. Fault management helps in detecting and troubleshooting the faults that cause disruptions in network services. Network elements contain monitoring and diagnostic tools for detecting various types of fault situations. Each fault is represented as an alarm and this alarm is send to NetAct.

NetAct Monitor (Application) with versatile monitoring tools enables user to:

- Collect, process, store, and display alarm information from the network in real-time.
- Visualize the network topology.
- Detect and analyze faults in network elements.

(b) Optimization

The radio network optimization process usually takes place when the monitored performance drops below the set targets, when a periodical tuning task is to be started or when there is need to optimize the behavior of new network elements in the network. NetAct Optimizer provides the functionality for automated, metric based optimization of operational WCDMA, LTE and GSM, networks. It provides the functionality for viewing, analyzing and optimizing the actual network performance and capacity.

As shown in the figure 2.2 Measurements are used for analyzing the network and service performance development against set targets. A detailed analysis is

performed to find the reasons behind decreased performance and to select the right corrective actions. In this phase, the relations between performance indicators and element parameters are analyzed. After the analysis phase, the configuration parameter settings are optimized and the set quality criteria are checked.

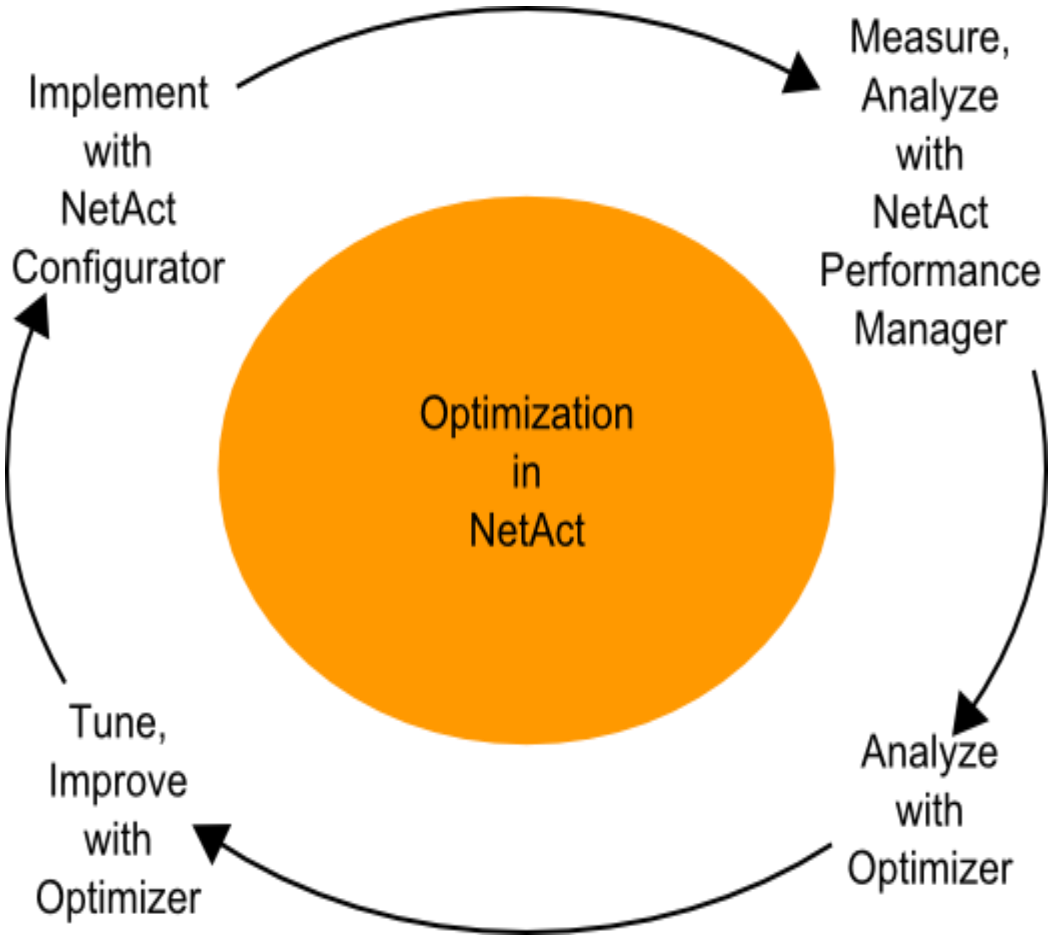

Figure 2.2 Optimization cycle in NetAct

When the corrections are verified and implemented into the network, the quality monitoring cycle depicted in figure starts from the beginning.

(c) Configuration management:

NetAct configuration management is used for configuring and managing radio and core network. User can make, for example single parameter changes, manage hardware data or launch new technologies and services. User can streamline and automate user configuration management processes and tasks, and react to network changes and demands.

With the NetAct Configurator applications, user can perform network-wide configuration management operations. NetAct Configurator provides one centralized data storage and the same tools and processes for all configuration management operations.

(d) Performance management:

The performance management functionalities in NetAct are made up of an extended family of applications for processing, analyzing, and visualizing performance data that is coming from different sources. Performance Management in NetAct gives a view of the network and service performance and makes it possible to analyze network data, create reports based on the data and distribute the information within the operator’s organization. Performance Manager is multi-vendor capable and collects data from the entire network that consists of network elements. It processes and stores the data, from days to years, depending on your requirements, and includes so several ready-made report packages.

The functional architecture of Performance Manager can be divided into 3 layers as in Figure 2.3:

• Integration layer

• Process layer

• Reporting layer

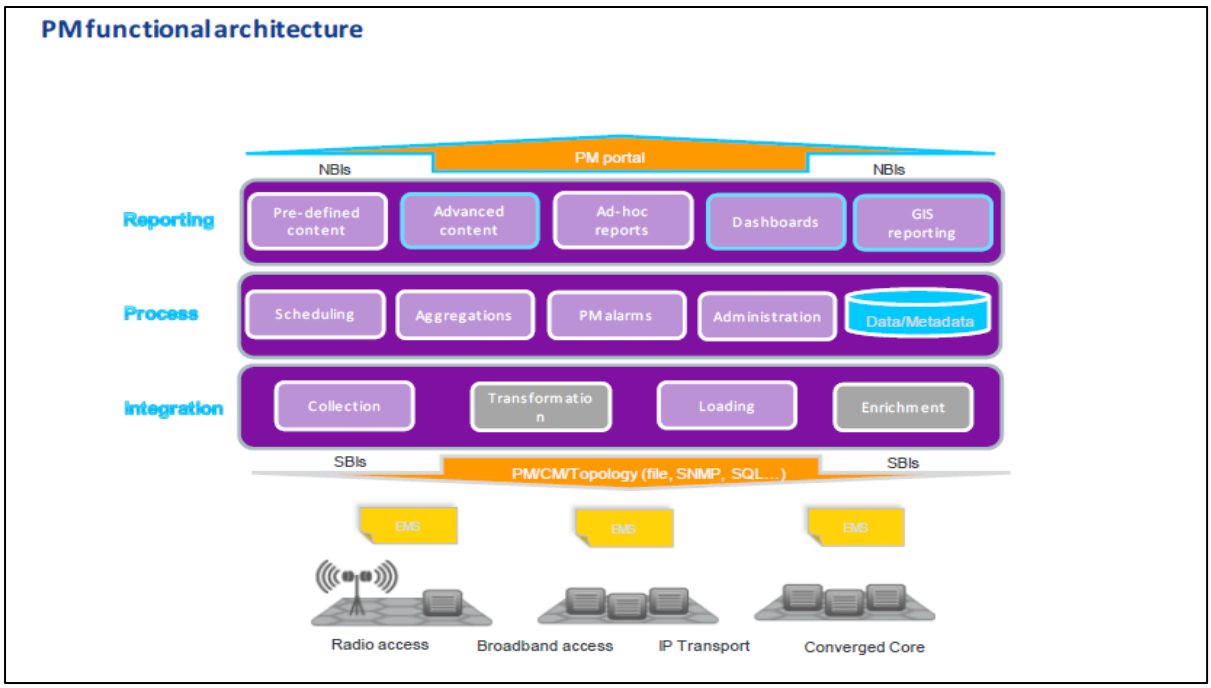

Figure 2.3 Performance Management in NetAct

The integration layer is responsible for all ETL (extraction, transformation and loading) related activities:

- Transformation adapts the vendor specific formats and protocols to PM internal xml-based format (OMeS).
- Collection uses mediation components that can connect to the integrated EMS(element management system), e.g., via ftp/sftp/SNMP, and collect PM/CM/Topology data. Push operations can also be used.
- Loading takes the PM data in OMeS format and stores it in the data warehouse.
- Enrichment takes care of managing & storing other types of data that can be used together with PM data (CM/FM statistics, GIS coordinates, etc.) on the reporting layer.

The process layer takes care of data preparation and summarization:

- Scheduling enables the administrator to increase system efficiency, by scheduling most heavy reports during offline hours.

- Raw data is aggregated in object/time dimension and stored according to customer preferences, allowing long-term and fast reporting. BH (Busy Hour) calculations from several algorithms are also provided.
- PM alarms can be generated based on KPI thresholds and/or KPI profiling deviations, using Thresholder & Profiler tool.
- Administration tools includes toolkits that allows to e.g., monitor system health and act upon abnormal behavior, change system configuration, troubleshoot data availability issues or even extend default data models.
- Data warehouse is based on Oracle database (w/ RAC option on project basis) and is used to store raw/aggregated PM data, pre-calculated KPI Values and other relevant information (e.g., topology, CM statistics, etc.). Metadata is stored in DW or file system and transversely serves all system tools, from mediation to reporting layer.

The reporting layer includes:

- Pre-defined content packages incorporating NSN telecommunications experience and know-how, in the form of useful KPIs and reports for major technology areas.
- Advanced Content packages providing valuable vendor and technology agnostic KPIs and dashboards.
- Ad-hoc reports, which can be created out of regular network counters, pre-defined or custom made KPIs.
- Dashboard framework, which allows users to tailor their own reports, with focus on presentation and layout capabilities.
- GIS reporting introduces specific map views, where meaningful KPIs and objects can be represented in geographical way

(e) Security management

Security management covers all the aspects shown in the Figure 2.4. Security management consists of user security, data security, software security, and network security. It is based on NSN security policy and guidelines that consist of customer requirements and international security standards.

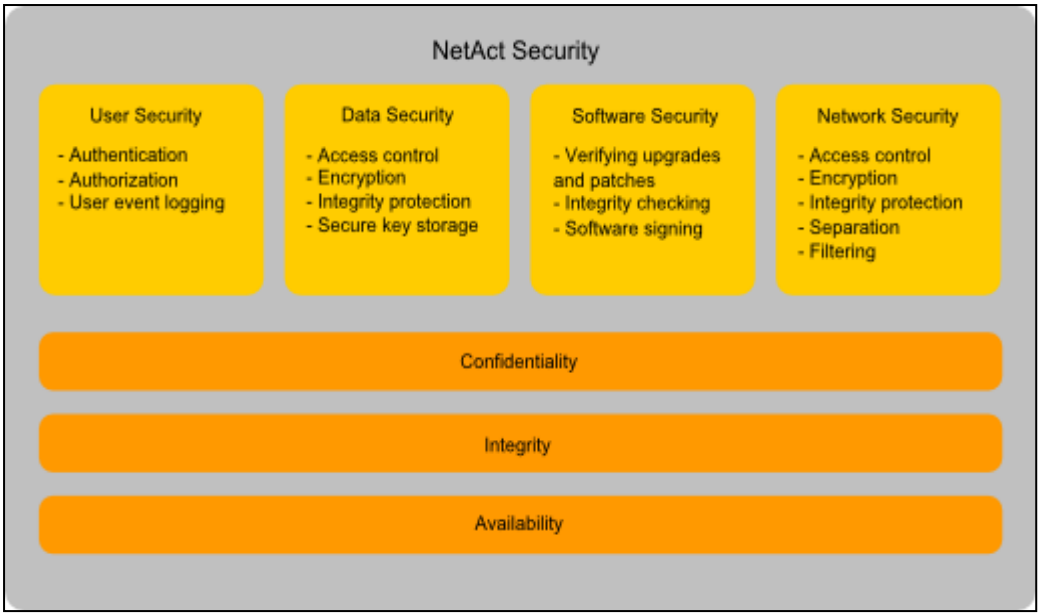

Figure 2.4 Security management

All security parts (user security, data security, software security, network security) are needed to guarantee the confidentiality, integrity and availability of the network.

Confidentiality means ensuring that information is accessible only to those authorized to have access and that the access rights remain uncompromised.

Integrity means the accuracy and consistency of information, and ensuring that the information has not been changed without proper authorization and any changes to the information can be verified.

Availability means the accessibility of information for all authorized parties in a timely manner.

(f) Network administration:

The network administration area of functionality of NetAct comprises tools for centralized network element software management, for backup and restore across network domains and multi-vendor systems, and for managing both network element and NetAct software licenses. The applications are designed to help to deploy new software releases, security patches, enhancements to existing releases, and fault fixes.

(g) NetAct administration:

As a network management system designed to operate and maintain a complex telecommunications network, NetAct needs to be up and running always. NetAct administration is essentially managing system reliability that is provided by the hardware, operating systems, third-party software, services, tools, interfaces, and applications comprising NetAct.

In NetAct, system reliability is ensured through the following basic features:

- High availability in the hardware configuration, virtual infrastructure, and select software components to eliminate single points of failure in power, disk, memory, CPU, network connectivity, virtual machine, or service availability.
- Load balancing, which takes care of the distribution of server load within a Web Sphere Application Server cluster and the distribution of resources within the virtual infrastructure.
- Online and offline backup and recovery solutions using vSphere Data Protection and the NetAct Backup Tool.
- System Self-monitoring for internal fault and performance management.
- Preventive Health Check, which is a tool that verifies the NetAct System status.

NetAct administration consists of a set of proprietary as well as third-party tools and instructions for the administrators of the NetAct system. NetAct system administrators manage the NetAct hardware, which consists of three main items: servers, storage devices, and network equipment such as switches and routers. Along with this role, they administer the virtual infrastructure and its corresponding resources. They also maintain the services that provide the run-time environment to manage network elements. The components that need to be managed include basic services (such as the directory service, database service, and domain name service to name a few) as well as southbound and northbound interfaces.

2.2 Time Series

Imagine you wanted to know how the temperature outside changes throughout the day. Once every hour, you'd check the thermometer and write down the time along with the current temperature. After a while, you'd have something like this:

Time	Value
09:00	24°C
10:00	26°C
11:00	27°C

Temperature data like this is one example of what we call a **time series**, a sequence of measurements, ordered in time. Every row in the table represents one individual measurement at a specific time.

Tables are useful when you want to identify individual measurements but make it difficult to see the big picture. A more common visualization for time series is the **graph**, which instead places each measurement along a time axis. Visual representations like the graph make it easier to discover patterns and features of the data that otherwise would be difficult to see.

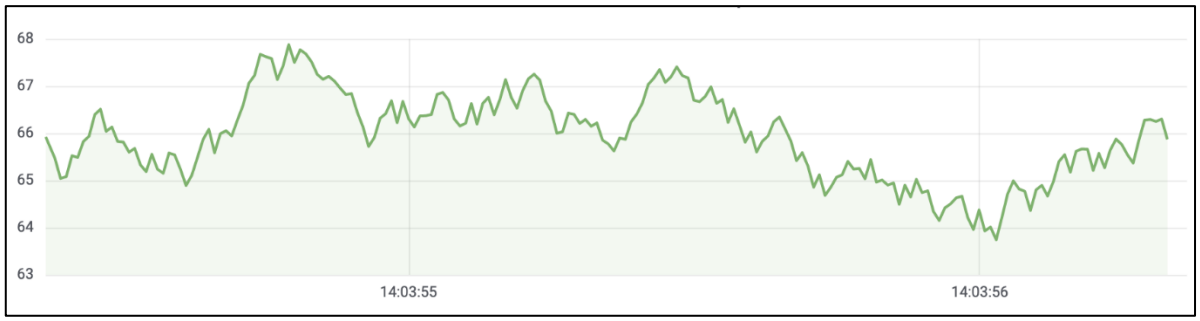

Figure 2.5. Time series data

Temperature data like the one in the example, is far from the only example of a time series. Other examples of time series are:

- CPU and memory usage
- Sensor data
- Stock market index

While each of these examples are sequences of chronologically ordered measurements, they also share other attributes:

New data is appended at the end, at regular intervals, for example, hourly at 09:00, 10:00, 11:00, and so on. Measurements are seldom updated after they were added, for example, yesterday's temperature doesn't change.

Time series are powerful. They help you understand the past by letting you analyze the state of the system at any point in time. Time series could tell you that the server crashed moments after the free disk space went down to zero.

Time series can also help you predict the future, by uncovering trends in your data. If the number of registered users has been increasing monthly by 4% for the past few months, you can predict how big your user base is going to be at the end of the year.

Some time series have patterns that repeat themselves over a known period. For example, the temperature is typically higher during the day, before it dips down at night. By identifying these periodic, or **seasonal**, time series, you can make confident predictions about the next period. If we know that the system load peaks every day around 18:00, we can add more machines right before.

Aggregating time series

Depending on what you're measuring, the data can vary greatly. What if you wanted to compare periods longer than the interval between measurements? If you'd measure the temperature once every hour, you'd end up with 24 data points per day. To compare the temperature in August over the years, you'd have to combine the 31 times 24 data points into one.

Combining a collection of measurements is called **aggregation**. There are several ways to aggregate time series data. Here are some common ones:

Average returns the sum of all values divided by the total number of values.

Min and **Max** return the smallest, and largest value in the collection.

Sum returns the sum of all values in the collection.

Count returns the number of values in the collection.

For example, by aggregating the data in a month, you can determine that August 2017 was, on average, warmer than the year before. Instead, to see which month had the highest temperature, you'd compare the maximum temperature for each month.

How you choose to aggregate your time series data is an important decision and depends on the story you want to tell with your data. It's common to use different aggregations to visualize the same time series data in different ways.

Time series and monitoring

In the IT industry, time series data is often collected to monitor things like infrastructure, hardware, or application events. Machine-generated time series data is typically collected with short intervals, which allows you to react to any unexpected changes, moments after they occur. As a consequence, data accumulates at a rapid pace, making it vital to have a way to store and query data efficiently. As a result, databases optimized for time series data have seen a rise in popularity in recent years.

Time series databases

A time series database (TSDB) is a database explicitly designed for time series data. While it's possible to use any regular database to store measurements, a TSDB comes with some useful optimizations.

Modern time series databases take advantage of the fact that measurements are only ever appended, and rarely updated or removed. For example, the timestamps for each measurement change very little over time, which results in redundant data being stored.

Collecting time series data

Now that we have a place to store our time series, how do we actually gather the measurements? To collect time series data, you'd typically install a **collector** on the device, machine, or instance you want to monitor. Some collectors are made with a specific database in mind, and some support different output destinations.

A collector either **pushes** data to a database or lets the database **pull** the data from it. Both methods come with their own set of pros and cons:

	Pros	Cons
Push	Easier to replicate data to multiple destinations.	The TSDB has no control over how much data gets sent.
Pull	Better control of how much data that gets ingested, and its authenticity.	Firewalls, VPNs or load balancers can make it hard to access the agents.

Since it would be inefficient to write every measurement to the database, collectors pre-aggregate the data and write to the time series database at regular intervals.

There are many other kinds of time-series data. To name a few: DevOps monitoring data, mobile/web application event streams, industrial machine data, scientific measurements.
These datasets primarily have 3 things in common:

- The data that arrives is almost always recorded as a new entry
- The data typically arrives in time order
- Time is a primary axis (time-intervals can be either regular or irregular)

Why do we need a time-series database?

(1) scale and (2) usability.
Scale: Time-series data accumulates very quickly, and normal databases are not designed to handle that scale. Relational databases fare poorly with very large datasets; NoSQL databases fare better at scale, but can still be outperformed by a database fine-tuned for time-series data. In contrast, time-series databases (which can be based on relational or NoSQL databases) handle scale by introducing efficiencies that are only possible when you treat time as a first-class citizen. These efficiencies result in performance improvements, including higher ingest rates, faster queries at scale (although some support more queries than others), and better data compression.

Usability: TSDBs also typically include functions and operations common to time-series data analysis such as data retention policies, continuous queries, flexible time aggregations, etc. Even if scale it not a concern at the moment (e.g., if you are just starting to collect data), these features can still provide a better user experience and make your life easier.

A **histogram** is a graphical representation of the distribution of numerical data. It groups values into buckets (sometimes also called bins) and then counts how many values fall into each bucket.

Instead of graphing the actual values, histograms graph the buckets. Each bar represents a bucket, and the bar height represents the frequency (such as count) of values that fell into that bucket's interval.

Histogram example

This histogram shows the value distribution of a couple of time series. You can easily see that most values land between 240-300 with a peak between 260-280.

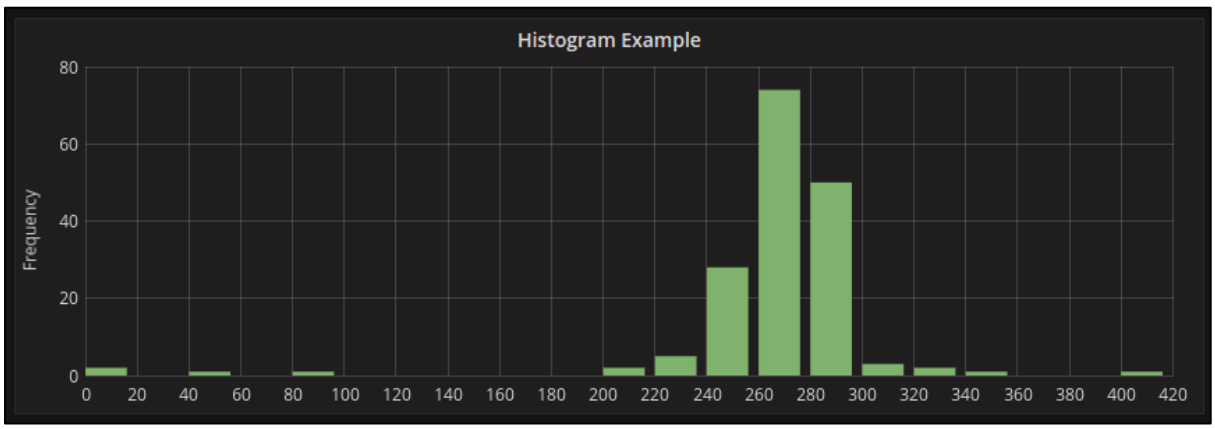

Figure 2.6 Histogram in Grafana

Histograms only look at **value distributions** over a specific time range. The problem with histograms is you cannot see any trends or changes in the distribution over time. This is where heatmaps become useful.

Heatmaps

A **heatmap** is like a histogram, but over time where each time slice represents its own histogram. Instead of using bar height as a representation of frequency, it uses cells and colors the cell proportional to the number of values in the bucket.

In this example, you can clearly see what values are more common and how they trend over time.

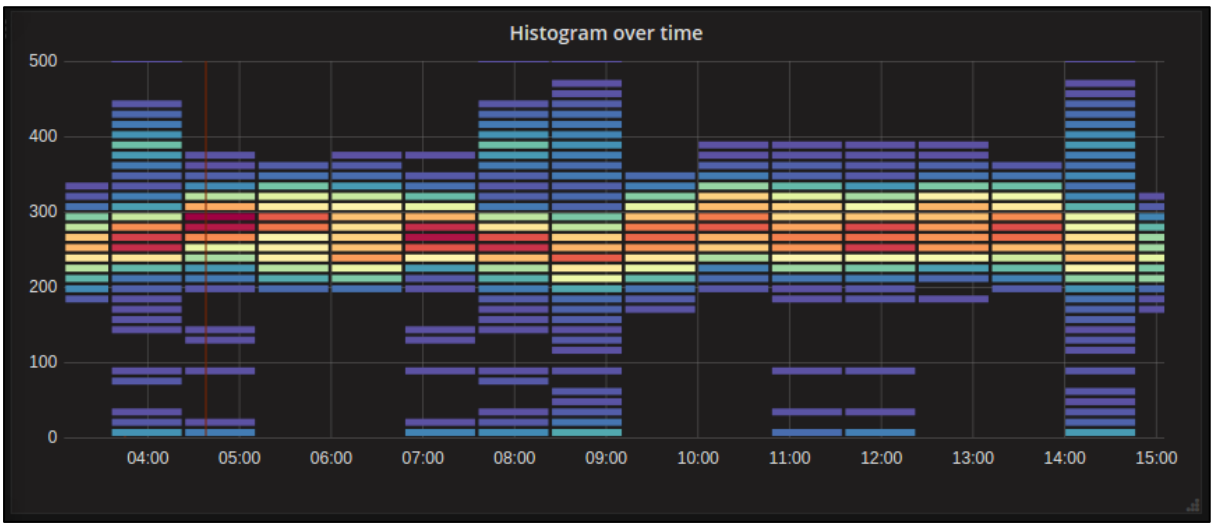

Figure 2.7 Heatmaps in Grafana

Pre-bucketed data

There are a number of data sources supporting histogram over time like Elasticsearch or Prometheus, but generally, any data source could be used if it meets the requirement, returns series with names representing bucket bound or returns series sorted by the bound in ascending order.

Raw data vs aggregated

If you use the heatmap with regular time series data (not pre-bucketed), then it’s important to keep in mind that your data is often already aggregated by your time series backend. Most time series query do not return raw sample data but include a group by time interval or maxDataPoints limit coupled with an aggregation function (usually average).

This all depends on the time range of your query of course. But the important point is to know that the histogram bucketing that Grafana performs might be done on already aggregated and averaged data. To get more accurate heatmaps it is better to do the bucketing during metric collection or store the data in Elasticsearch, or in the other data source which supports doing histogram bucketing on the raw data.

If you remove or lower the group by time in your query to return more data points your heatmap will be more accurate but this can also be very CPU and memory taxing for your browser and could cause hangs and crashes if the number of data points becomes unreasonably large.

Figure 2.8 NetAct timeseries data in Grafana

2.3 Zabbix

Zabbix is an enterprise-class open source distributed monitoring solution. It is a software that monitors numerous parameters of a network and the health and integrity of servers. Zabbix uses a flexible notification mechanism that allows users to configure e-mail-based alerts for virtually any event. This allows a fast reaction to server problems. It also offers excellent reporting and data visualization features based on the stored data. This makes Zabbix ideal for capacity planning.

Zabbix supports both polling and trapping. All Zabbix reports and statistics, as well as configuration parameters, are accessed through a web-based front end. A web-based front end ensures that the status of your network and the health of your servers can be accessed from any location. Properly configured, Zabbix can play an important role in monitoring IT infrastructure. This is equally true for small organizations with a few servers and for large companies with a multitude of servers.

Zabbix offers features like:

- Auto-discovery of servers and network devices.
- Distributed monitoring with centralized WEB administration.
- Support for both polling and trapping mechanisms.
- Server software for Linux, Solaris, HP-UX, AIX, Free BSD, Open BSD, OS X
- Native high-performance agents (client software for Linux, Solaris, HP-UX, AIX, Free BSD, Open BSD, OS X, Tru64/OSF1, Windows NT4.0, Windows 2000, Windows 2003, Windows XP, Windows Vista)
- Agent-less monitoring.
- Secure user authentication.
- Flexible user permissions.
- Flexible e-mail notification of predefined events.
- High-level (business) view of monitored resources.

2.3.1 Definitions

(a) Zabbix Server

This is the center of the Zabbix software as shown in Figure 2.5. The Server can remotely check networked services (such as web servers and mail servers) using simple service checks, but it is also the central component to which the Agents will report availability and integrity information and statistics.

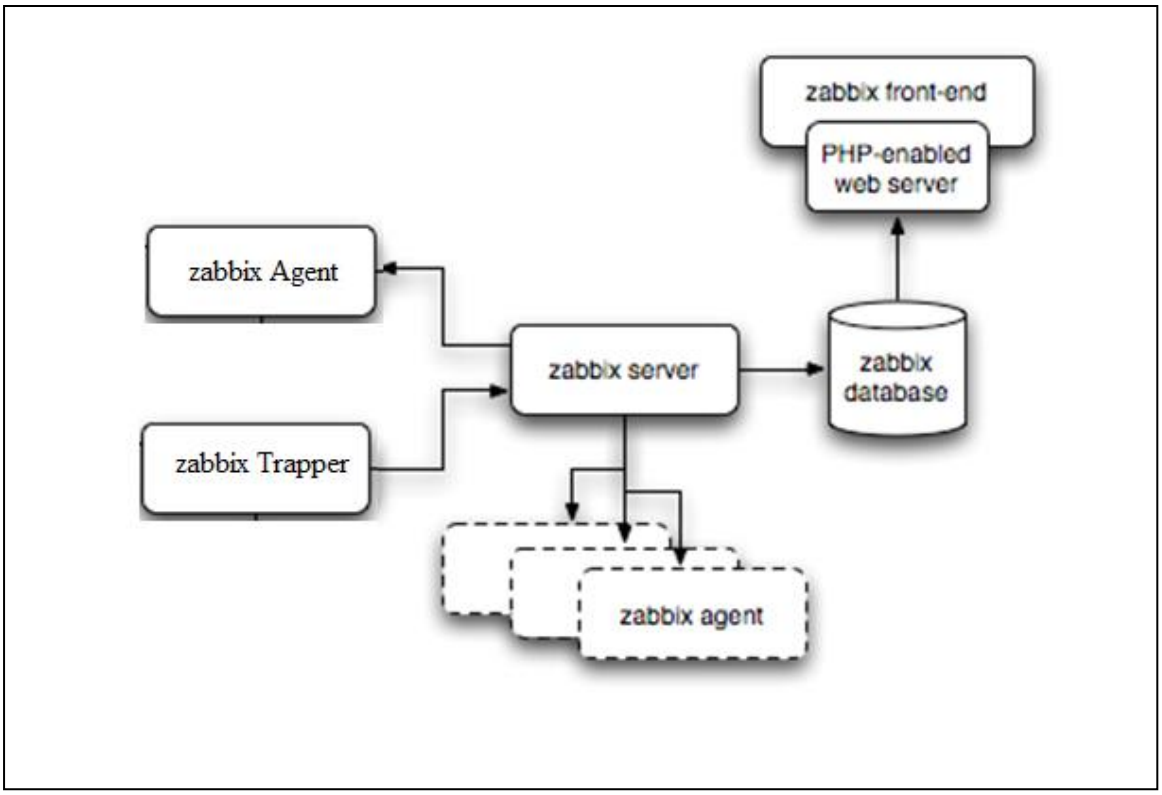

Figure 2.9 Zabbix Architecture

The Server is the central repository in which all configurations, statistical and operational data are stored, and it is the entity in the Zabbix software that will actively alert administrators when problems arise in any of the monitored systems. Zabbix can also perform agent-less monitoring and also monitor network devices using SNMP agents.

(b) Zabbix Proxy

The Proxy is an optional part of Zabbix deployment. The Proxy collects performance and availability data on behalf of Zabbix Server. All collected data is buffered locally and transferred to Zabbix Server the Proxy belongs to. Zabbix Proxy is an ideal solution for a centralized monitoring of remote locations, branches, networks having no local administrators. Zabbix Proxies can also be used to distribute load of a single Zabbix Server. In this case, only Proxies collect data thus making processing on the server less CPU and disk I/O hungry.

(c) Zabbix Agent

In order to actively monitor local resources and applications (such as hard drives, memory, processor statistics etc.) on networked systems, those systems must run the Zabbix Agent. The Agent will gather operational information from the system on which it is running, and report these data to the Zabbix for further processing. In case of failures (such as a hard disk running full, or a crashed service process), the Zabbix Server can actively alert the administrators of the particular machine that reported the failure. The Zabbix Agents are extremely efficient because of use of native system calls for gathering statistical information.

(d) Zabbix front-end (Web Interface)

In order to allow easy access to the monitoring data and then configuration of Zabbix from anywhere and from any platform, the Web-based Interface is provided. The Interface is a part of the Zabbix Server, and is usually (but not necessarily) run on the same physical machine as the one running the Zabbix Server.

(e) Zabbix Features

Zabbix is a highly integrated network monitoring solution, offering a multiplicity of features in a single package.

Data gathering

- availability and performance checks support for SNMP (both trapping and polling), IPMI, JMX, VMware monitoring
- custom checks
- gathering desired data at custom intervals
- performed by server/proxy and by agents

Flexible threshold definitions

- you can define very flexible problem thresholds, called triggers, referencing values from the backend database

Highly configurable alerting

- sending notifications can be customized for the escalation schedule, recipient, media type notifications can be made meaningful and helpful using macro variables
- automatic actions include remote commands

Real-time graphing

- monitored items are immediately graphed using the built-in graphing functionality

Web monitoring capabilities

- Zabbix can follow a path of simulated mouse clicks on a web site and check for functionality and response time

Extensive visualisation options

- ability to create custom graphs that can combine multiple items into a single view
- network maps
- custom screens and slide shows for a dashboard-style overview
- reports
- high-level (business) view of monitored resources

Historical data storage

- data stored in a database
- configurable history
- built-in housekeeping procedure

Easy configuration

- add monitored devices as hosts
- hosts are picked up for monitoring, once in the database
- apply templates to monitored devices

Use of templates

- grouping checks in templates
- templates can inherit other templates

Network discovery

- automatic discovery of network devices
- agent auto registration
- discovery of file systems, network interfaces and SNMP OIDs

Fast web interface

- a web-based frontend in PHP
- accessible from anywhere
- you can click your way through
- audit log

Zabbix API

- Zabbix API provides programmable interface to Zabbix for mass manipulations, 3rd party software integration and other purposes.

Permissions system

- secure user authentication
- certain users can be limited to certain views

Full featured and easily extensible agent

- deployed on monitoring targets
- can be deployed on both Linux and Windows

Binary daemons

- written in C, for performance and small memory footprint
- easily portable

Ready for complex environments

- remote monitoring made easy by using a Zabbix proxy

The Analysis and Reporting System is an amalgamation of the online-oriented Performance monitoring and the offline-oriented Performance Reporting as shown in Figure 2.6. This System is deployed on Zabbix server and accessed from a web-client.

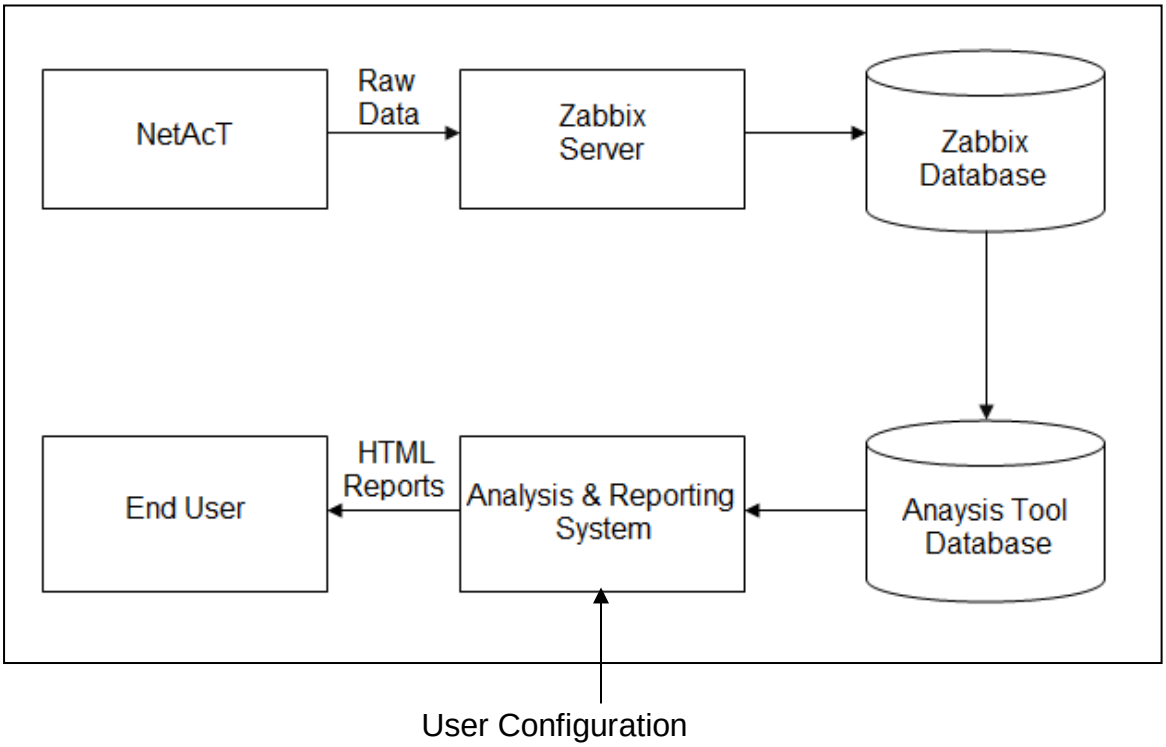

Figure 2.10 Analysis and Reporting System in Zabbix

It is a two-tier architecture with one-to-one mapping between the Zabbix database and the system's database. Zabbix collects the network wide data from the NetAct into the database. This raw data is fetched from Zabbix Database and the configuration information is provided as a user input to perform the required analysis and depending on the customer requirements, Testcases generate the final analyzed report for the end User. The results from this report provide measured and analyzed performance characteristics of the NetAct.

2.4 GRAFANA

This topic provides a high-level look at Grafana, the Grafana process, and Grafana features. It's a good place to start if you want to learn more about Grafana software.

Grafana is an open source visualization tool that can be used on top of a variety of different data stores but is most commonly used together with Graphite, InfluxDB, Prometheus, Elasticsearch and Logz.io. As it so happens, Grafana began as a fork of Kibana, trying to supply support for metrics (a.k.a. monitoring) that Kibana (at the time) did not provide much if any such support for.

Essentially, Grafana is a feature-rich replacement for Graphite-web, which helps users to easily create and edit dashboards. It contains a unique Graphite target parser that enables easy metric and function editing. Users can create comprehensive charts with smart axis formats (such as lines and points) as a result of Grafana's fast, client-side rendering — even over long ranges of time — that uses Flot as a default option.

Grafana is open source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored. In plain English, it provides you with tools to turn your time-series database (TSDB) data into beautiful graphs and visualizations.

For example, if you want to view weather data and statistics about your smart home, then you might create a playlist. If you are the administrator for a corporation and are managing Grafana for multiple teams, then you might need to set up provisioning and authentication.

Explore metrics and logs

Explore your data through ad-hoc queries and dynamic drilldown. Split view and compare different time ranges, queries and data sources side by side.

Alerts

If you're using Grafana alerting, then you can have alerts sent through a number of different alert notifiers including PagerDuty, SMS, email, VictorOps, OpsGenie, or Slack.

Alert hooks allow you to create different notifiers with a bit of code if you prefer some other channels of communication. Visually define alert rules for your most important metrics.

Annotations

Annotate graphs with rich events from different data sources. Hover over events to see the full event metadata and tags.

This feature, which shows up as a graph marker in Grafana, is useful for correlating data in case something goes wrong. You can create the annotations manually—just control-click on a graph and input some text—or you can fetch data from any data source.

Dashboard variables

Template Variabled allow you to create dashboards that can be reused for lots of different use cases. Values aren't hard-coded with these templates, so for instance, if you have a production server and a test server, you can use the same dashboard for both.

Templating allows you to drill down into your data, say, from all data to North America data, down to Texas data, and beyond. You can also share these dashboards across teams within your organization—or if you create a great dashboard template for a popular data source, you can contribute it to the whole community to customize and use.

Configure Grafana

Configuration covers both config files and environment variables. You can set up default ports, logging levels, email IP addresses, security, and more.

Authentication

Grafana supports different authentication methods, such as LDAP and OAuth, and allows you to map users to organizations.

In Grafana Enterprise, you can also map users to teams: If your company has its own authentication system, Grafana allows you to map the teams in your internal systems to teams in Grafana. That way, you can automatically give people access to the dashboards designated for their teams.

Provisioning

While it's easy to click, drag, and drop to create a single dashboard, power users in need of many dashboards will want to automate the setup with a script. You can script anything in Grafana.

For example, if you're spinning up a new Kubernetes cluster, you can also spin up a Grafana automatically with a script that would have the right server, IP address, and data sources preset and locked in so users cannot change them. It's also a way of getting control over a lot of dashboards.

Permissions

When organizations have one Grafana and multiple teams, they often want the ability to both keep things separate and share dashboards. You can create a team of users and then set permissions on folders, dashboards, and down to the data source level if you're using Grafana Enterprise.

2.5. System Performance and Dimensioning Tool

The System Performance team is responsible for executing various test-cases and test-environments in simulation before deploying the build/version/upgrade to a customer. This teams ensures that there are no escape defects from R&D development to customers. These tests ensure that the customer faces no issues when they deploy NetAct.

There are two scenarios in deploying NetAct to customers site:

1. Greenfield deployment
2. Network Expansion

Greenfield: Greenfield deployment is where a completely new Network along with NetAct is deployed in the customer site. This is when a customer would opt to use NetAct where deployment of a network is done where none existed before.

A customer would provide the network details and they would like to use NetAct as their NMS and EMS tool. NetAct support teams will estimate the overall configuration details required to setup the network by obtaining the customer requirements. Since there are three primary configurations in NetAct, using primary KPIs estimation and the dimensioning tool, the NetAct support teams can provide customer with the optimal type of supported configuration.

Network Expansion: When customers wants to increase their existing capacity by adding new Network elements due to the new users, NetAct also needs to be planned for increase in capacity to support additional Network elements, then NetAct support teams approaches NetAct for increase in the supported capacity (This includes increase in Network Elements, Network technologies, tools and other hardware elements).

The purpose of the project is to perform exploratory data analysis for different use - cases on these test results and tabulate the KPI data results. After tabulating the data of various KPIs for different use – cases, we implement a machine learning model to predict the configuration details for both greenfield deployment and capacity increase.

CHAPTER 3: DESIGN AND IMPLEMENTATION

This chapter discusses the details of all the use-cases in which exploratory data analysis, explanatory data analysis, visualization, predictive analysis using machine learning methods was implemented using data from network monitoring tools such as Grafana and Zabbix.

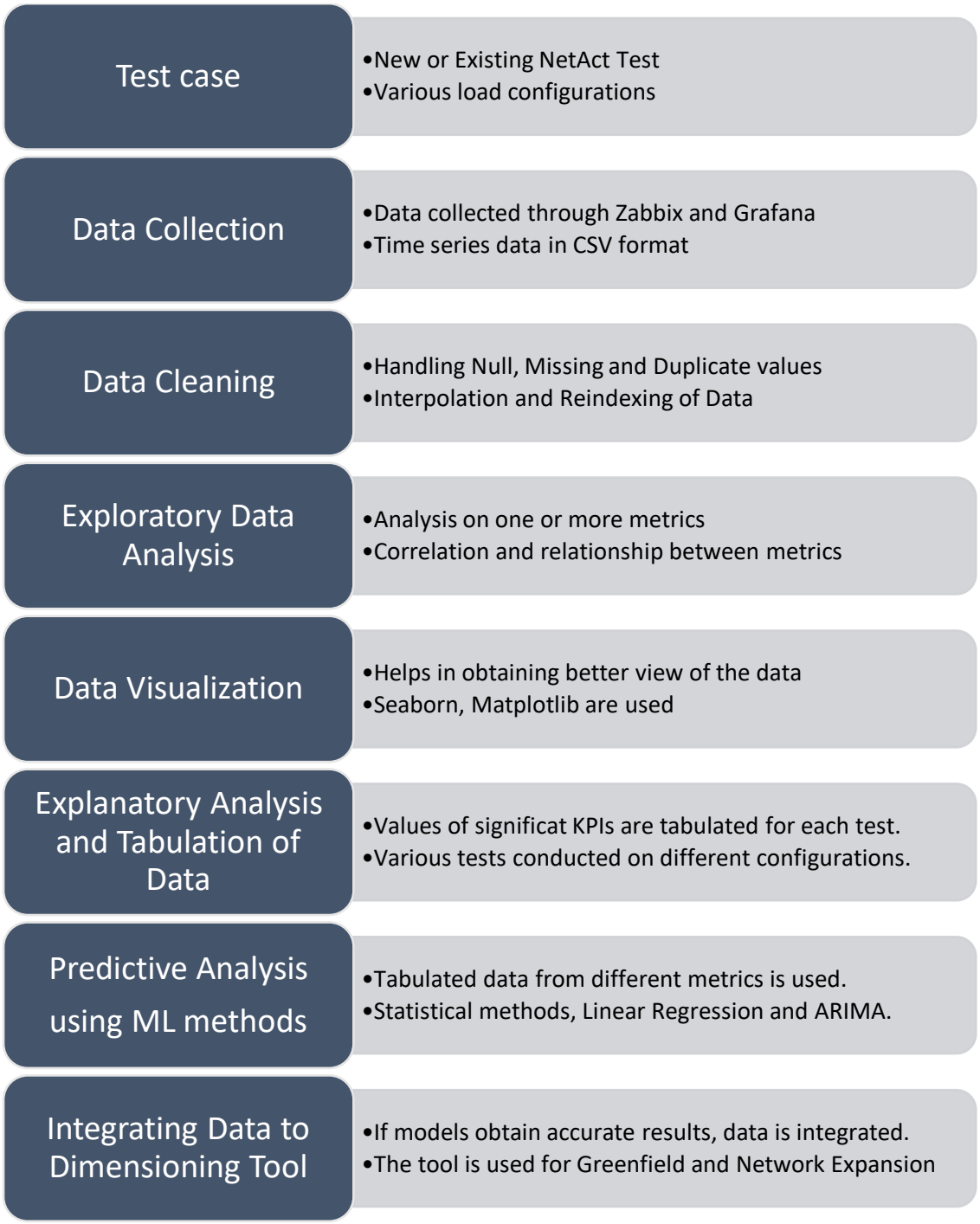

Figure 3.1: Workflow of use-cases in the project

The above workflow is implemented for all the use cases that are implemented in the project. Every use-case is specified based on the tests being conducted, load configurations and simulators being used at a particular time.

When a test is being conducted, a load model is built for the test where the details of metrics being tested is stored in the load model. Based on the load model and the configuration of the load model we determine the type of use case that needs to be implemented

The data is primarily collected from Grafana and Zabbix, SQL is also used to fetch the data directly from DCFW (Data Collection FrameWork) integrated to labs in NetAct. This data is in univariate time series format, the data needs further cleaning and preprocessing as the data consists null values, duplicated values and missing values. These can be handled using averaging, reindexing, interpolation with the help of forward and backward fill using the existing data.

The next step is to perform exploratory analysis and visualization of the collected data, find patterns, trends and anomalies in the tests and obtain inferences, using these inferences and data we tabulate the conclusions for each test with the resultant values of critical KPIs that impact the test.

The tabulated data is then used for predictive analysis by testing the tabulated data with machine learning models which are associated with numerical values and time series data. If the resultant values from predictive analysis are accurate and match with the respective test/load configuration data, the values are integrated or replaced in the dimensioning tool.

Here are the 5 use cases in the project:

1. Peak usage measure
2. Correlation analysis
3. Stepwise UI load tests and analysis
4. Raw and Aggregated data analysis
5. Time series analysis using ARIMA

4.1 PEAK USAGE MEASURE

There are several metrics which remain idle when there are no processes running but have some percent of usage which attributes to other dependent metrics of NetAct. The CPU of a DB Virtual Machine (VM) is dependent of aggregation of the data, CPU of PM (Performance Management) VM is dependent on generating reports, etc. These types of tasks are always running even when there are no prominent processes are running in NetAct system. The idle usage for any lab will vary from 0% to 20% and this usage will grow when prominent tasks are initiated.

The goal of this chapter is to measure the actual PM usage for a NetAct lab w.r.t different configurations and counters. The time interval at which the counters are being sent is the important factor that needs to be considered.

Working methodology:

If counters are being sent at 15-minute interval then the complete data is distributed into 4-time groups with 15-minute data points each which comes into 60 data points per hour. If counters are sent at 30 min interval, its 30 data points with 2 interval groups. Basically, if there are 15-minute counters being sent in a lab for a period of 24 hours, we have 24*4 = 96 groups with 1440 data points.

For each group we measure the number of minutes the CPU usage stays more than 20% whenever counters are being pumped into the VM. For example, group X has 15 min counters and CPU usage crosses 20% at 4^{th} minute and comes down below 20% at 11^{th} minute, so it stayed above 20% for 7 minutes. So, the exact number of minutes PM VM attributed to counters is 7 minutes in X group. We do this analysis for all the groups of data that has been collected, visualize the results and select the number of minutes with highest frequency.

Here is a visual representation of how to infer the results:

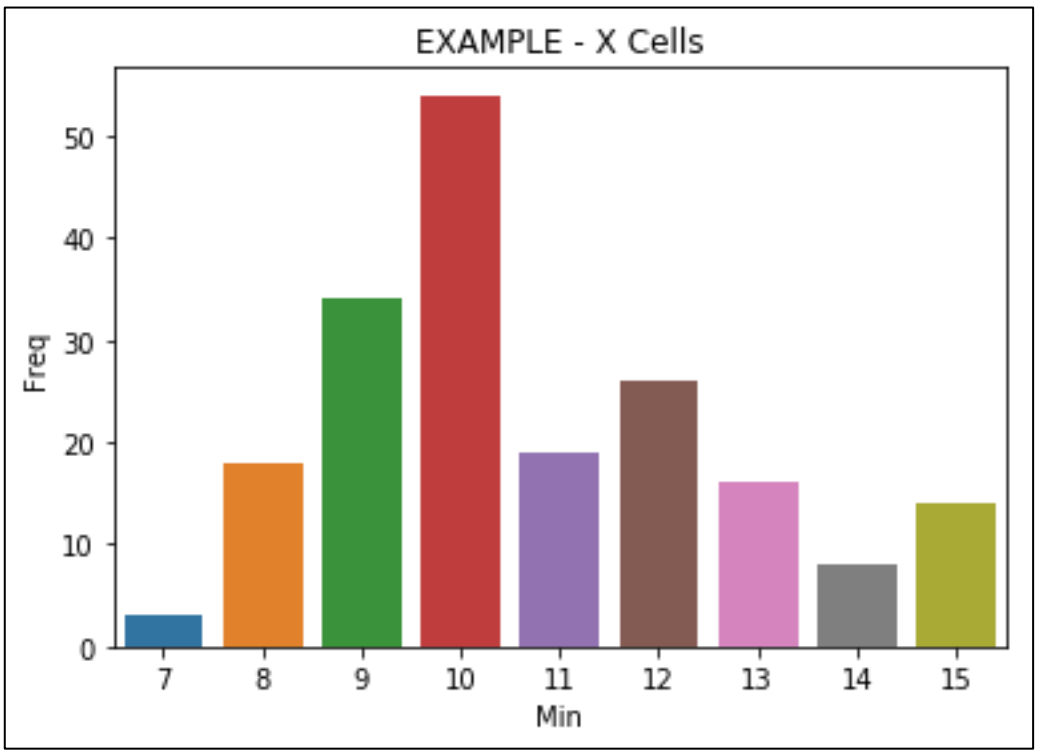

Figure 3.2: Measuring the Peak usage

- X-axis – Observed N^{th} Minute
- Y-axis – Number of times CPU was above 20% for N^{th} Minute

Inference: For X cells/counters at every 15-minute interval, the greatest number of times CPU stayed greater than 20% was 10 Minutes.

This test was conducted on various labs with different configurations and load and all the results were tabulated for comparison analysis. In the below example 3 different tests were conducted with increasing load in counters and the analysis was performed for all three tests on 4 PM VMs (VM 66, VM 67, VM68 and VM 69 in NetAct belongs to PM)

Results:

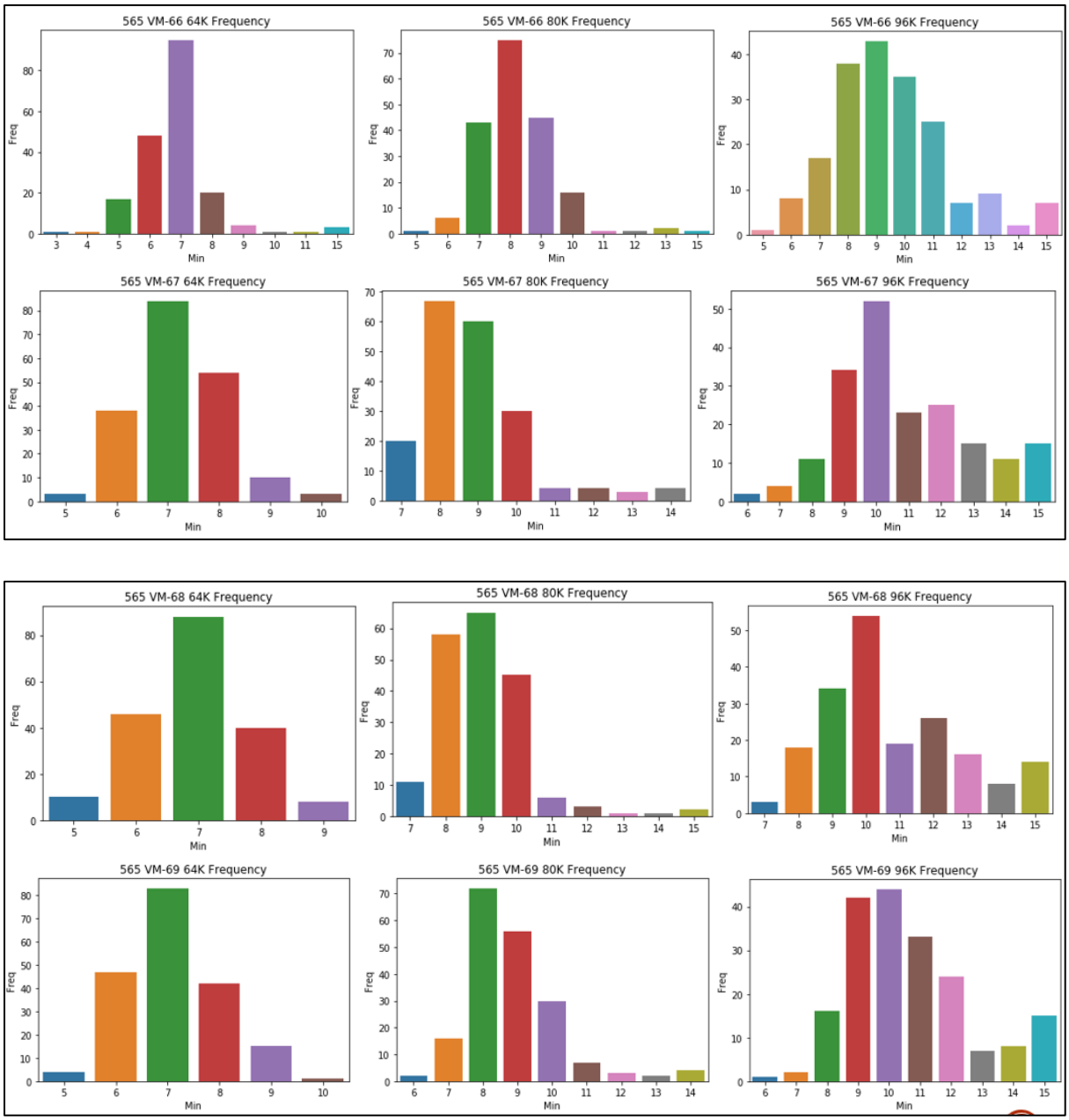

Figure 3.3: Peak usage for all four VMs in 565.

After doing the analysis of all the four PM VMs we tabulate the data to see the results. In these results we take the majority vote of peak usage among the PM VMs as the counters are distributed equally among each PM VM. We can see that the peak usage minutes keeps increasing with increase in overall load of tests conducted in the lab. Similar analysis was done on different labs with different NetAct configurations, load and resultant data was tabulated.

LAB: 565	**64K**	**80K**	**96K**
VM 66	7 min	8 min	9/8/10 min
VM 67	7 min	8/9 min	10 min
VM 68	7 min	9/8 min	10 min
VM 69	7 min	8/9 min	10/9 min
Peak Value	**7 MIN**	**8 MIN**	**10 MIN**

Results of 565 lab

When this analysis was conducted for others labs, smaller configuration labs with smaller load had a peak time of 1 to 3 minutes, mainstream and large labs had peak time of 4 to 8 minutes. The peak time depends on configuration of the lab as well as the load at which the lab is being tested. Sometimes the peak usage was greater than 20% for all the data points in an interval, i.e. the peak usage is always greater than 20% in all the 15 minutes of an interval.

When investigated I found that this is caused due to two factors:

- Due to Backup, Restore, Integration of Lab.
- The lab couldn’t pump all the counters within the interval time.

Conclusions:

- The amount of peak time unit (CPU>20%) always increased with increase in every cells.
- In some cases, the peak time unit had 2 influential time points for the same test scenario.
- There were also instances where the CPU Utilization was greater than 20% throughout the recorded interval (15 minutes).
- If all 4 PM VM’s have the similar results, any one VM can be considered for modelling.

4.2 CORRELATION ANALYSIS

In Grafana/Zabbix the data is presented in univariate time-series format. The X axis always represents the time-frame of the data, Y axis is the the change in value of metric over a recorded period of time. Hence it is not possible to estimate the dependence and correlation between metrics in the monitoring tools.

In this chapter, we collect multiple metrics over a same time period and perform correlation and statistical analysis of data. The process is to collect metrics from different labs over a time period and compare the statistics of the data.

WORKING METHODOLOGY:

When a lab is tested with a specific load configuration, it impacts on different metrics of the system. The metrics are captured and stored in DCFW; visual representation of this data is found in network monitoring tools. First the data for a specific time period is selected and then this data is obtained in CSV format, data of necessary metrics is collected for the same time period and analysis is done on the data. Similar process is done for different labs and load configurations.

There are two further use-cases on this:

1. Correlation and pair plot of data.
2. Disk Write Rate and IOPS analysis

1. Heatmaps and Pair plots

In this case the data is collected for a period of 30 days, the metrics involved in this analysis are PM (Performance Management) related, Counters (CTR), Files (FLS), Inserts (IST) and CPU usage of a PM VM (Virtual Machine).

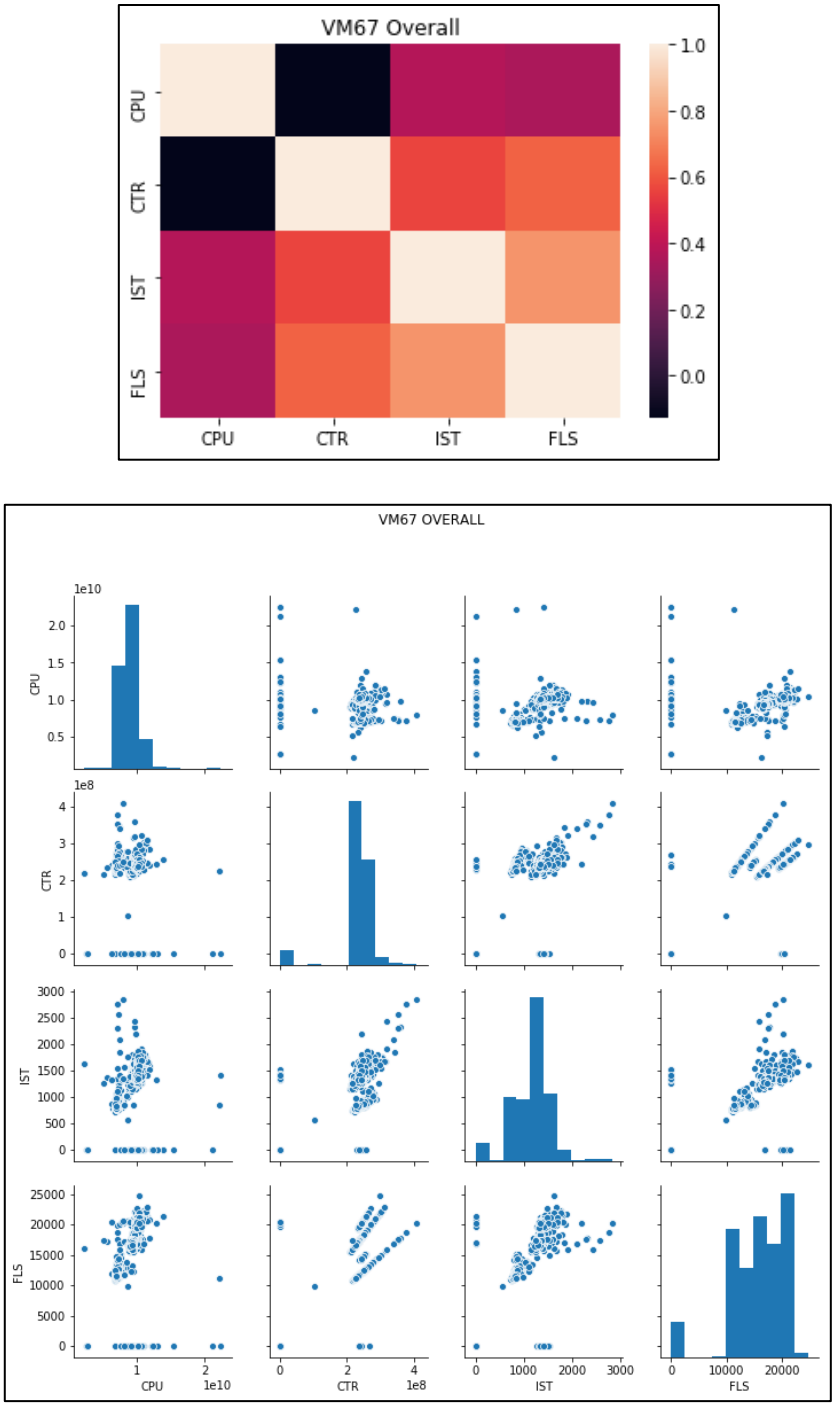

Figure 3.4: Correlation and pair plots of data.

From Fig 3.3 we can observe that the correlation between metrics is positve (>0.5) for Counters, Files and Inserts and there is very less or no correlation when these metrics are compared with CPU usage of the virtual machine.This is because the PM CPU

usage is less than 5-10% when there is no load or process running in the machine, this inturn impacts the overall correlation of the data.

Coming to the pair plots we can see the positive linear dependence among the PM metrics. If you carefully observe the CTR vs FLS in the pair plot we can see that there are 3 different linear points in the plot. When investigated the reason for this was the number of Files being sent were modified while the value of counters being sent remained unchanged.

Since there were 3 different splits, the data was further divided into 3 parts and further correlation analysis was done,

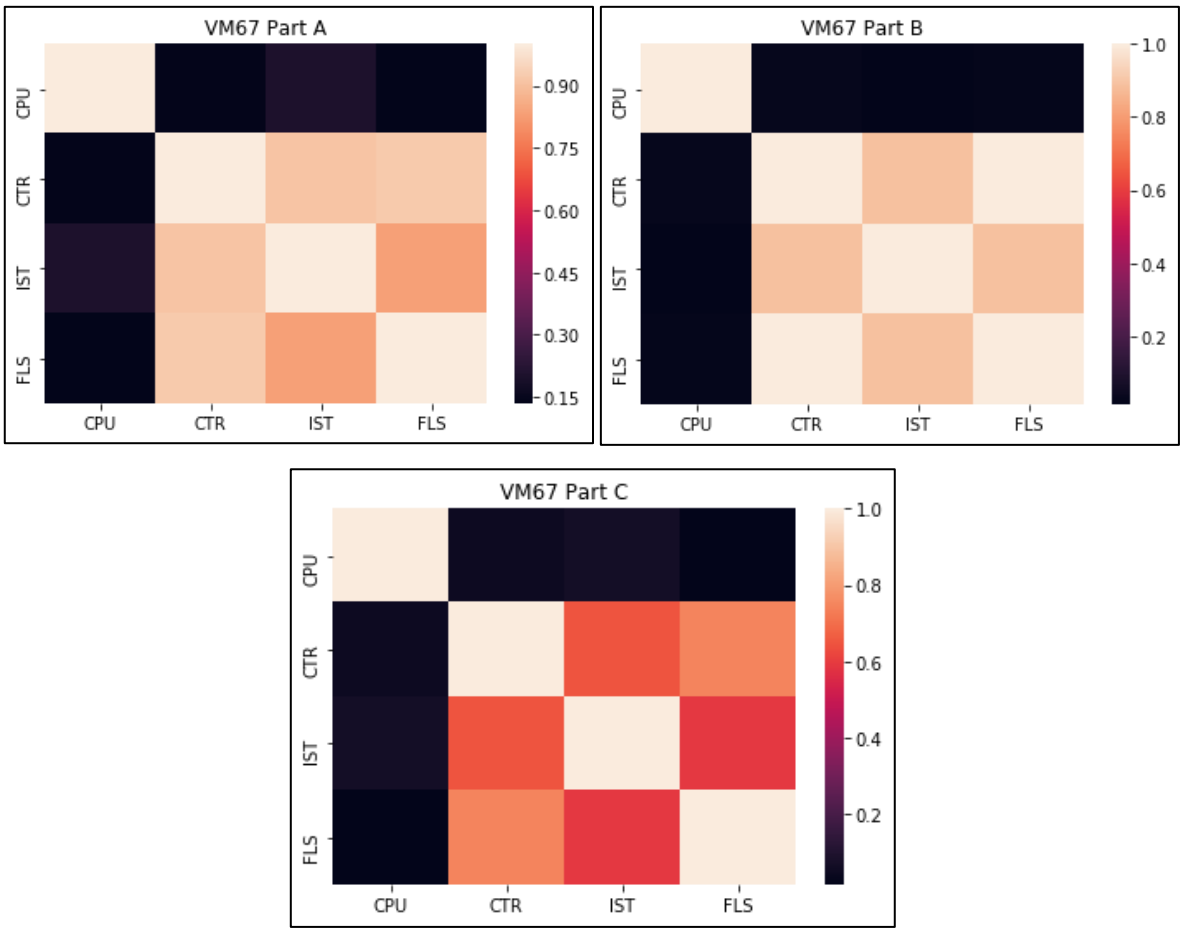

Figure 3.5 Correlation when Data is split into 3 parts

After dividing the data in splits we can see better correlation among metrics, however correlation between CPU and other metrics is still minimal. Obtaining heatmaps and pair plots of data will help nin understanding the correlation, distribution of data and

the significane of relationship among the metrics. This further helps in performing an effiecient exploratory analysis on the data.

2. Disk Write Rate and IOPS analysis

Here we analyze the change in IOPS and Disk Write Rate (DWR) due to the total number of counters that are being pumped into the lab.

The challenge is to relate the total number IOPS or Disk write rate with Counters as IOPS and DWR data is generated at 5-minute intervals and the counters data is generated at 60 min (hourly) intervals. Some options to interpolate this data is taking the average of all the 5-minute data points for each hour or taking the data point with maximum value at each hour or adding up all the 5-minute data points for each hour.

After trying out all these options taking the sum was the best and optimal approach, the data was collected for 3 different labs with all labs having different load on tests that were conducted.

Here are the details of the load and the resultant average IOPS an DWR data captured from Grafana:

LAB	**Counters**	**IOPS**	**DWR**
474	460 Mil	730	25MB
565	800 Mil	850	35MB
547	900 Mil	850-900	85MB

Scatter plot is the best approach to obtain the relationship among 2 metrics. Here we plot scatterplots with X-axis being the Counters (CTR) and one of the plots having Y-axis as the sum of 5-minute IOPS and the other plot with Y-axis as highest throughput at every hour.

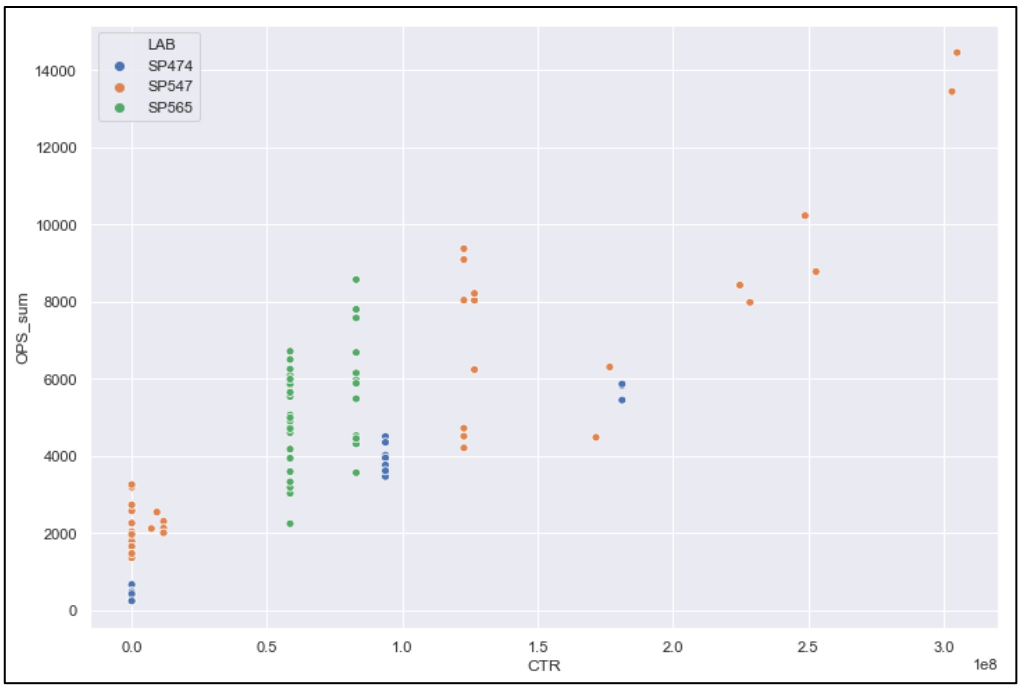

Figure 3.6: IOPS vs Counters

In Fig 3.6 we can observe that with increase in counters there is an increase in IOPS, this shows positive linear dependence in data. We see multiple counters values for same lab because the counters are being sent at different measurement intervals (15-minute, 30 minute and 60-minute intervals).

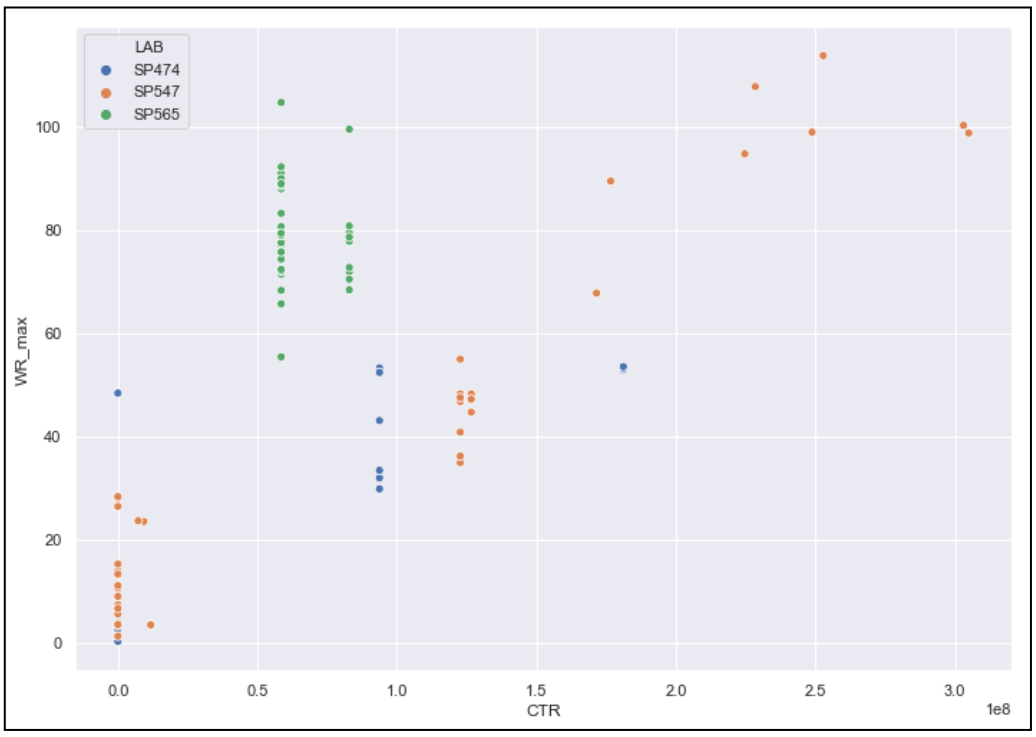

Figure 3.7: Disk write rate vs Counters

In Fig 3.7 we see that 547 lab has higher write rate even though the number of counters is less, this is because the counters are sent at both 15 and 30 minute intervals and the processing time for counters in lab is lesser than other labs which in turn results in higher write rate. However even in this plot we can see the linear dependence in terms of overall relationship between these metrics.

In both the cases we can see that counters have a positive correlation with IOPS and Disk write rate. Even in the first use case we could see that the metrics are correlated with each other with an exception of CPU usage. Although the challenge is to clean the data and transform or reindex various metrics at different time intervals into one single time interval, obtaining multiple metrics for same time period and performing exploratory data analysis will be very helpful in understanding the overall significance of the tests that are being performed in NetAct.

In this chapter we came to know that depicting the correlation, relationship and analyzing more metrics will in turn help for deducing better inferences and conclusions about the overall data.

4.3 STEPWISE UI LOAD TESTS AND ANALYSIS

NetAct Monitors help users to check various ongoing operations in a test. Some prominent operations which can be viewed and monitored are Alarm list, Alarm history, Alarm ACK and UNACK, Alarm Filter, PLMN (Public Land Mobile Network).

Whenever a user launches a monitor on client side or server side, it consumes some amount of RAM and Disk, with launch of a greater number of monitors the consumption of these resources keeps getting higher. At one point of time the resources will be consumed nearly at maximum capacity and when this happens the NetAct monitors will suffer from lag, delay in performing operations and end up crashing.

NetAct Monitor specifies a maximum capacity of 80 monitors for Large and XXL configuration at any point of time, It shows a warning message if an user tries to launch a monitor after the specified limit. Sometimes, even when the number of monitors launched are less than 80 they tend to have more latency, lag and in some cases they also end up crashing or having no response when any operation is performed.

The goal of this analysis was to check the behavior of monitors in different NetAct configurations (Small, Large, XXL), observe the CPU usage consumption, tabulate the overall time taken to perform various operations in the monitor.

Initially, CPU and Disk usage resources will be equally distributed for all the monitors that are being launched in the lab, but when these resources reach the threshold levels the resources will start getting prioritized. For example, if there are 10 monitors consuming 500 MB of memory each and overall RAM of the machine is 6 GB and CPU usage levels reach 90%, the 11^{th} monitor when launched will obtain resources from any one of the 10 monitors that are already launched in the lab.

CPU usage in different configurations

The tests were conducted on 3 different configuration labs:

- SP 439: Small configuration lab, Max monitors supported: 40
- SP 564: Large configuration lab, Max monitors supported: 80
- SP 636: XXL configuration lab, Max monitors supported: 80

Process of the stepwise tests:

- Start the test with 10 monitors, run for 6-8 hours period and obtain the minimum, average and maximum CPU usage for each hour in DB and WAS VM.
- Obtain single values for minimum, maximum and average usage by observing the obtained data points from the test run using averaging or majority vote method.
- Cleanup 10 monitors and launch 20 monitors, run the test for 6-8 hours, obtain min, max and avg values.
- Cleanup the 20 monitors.
- Continue the procedure stepwise by incrementing 10 monitors until all the 80 monitors are launched in the test.
- Tabulate the data of minimum, maximum and average CPU usage.

The above process is performed for all the 3 configurations and the results are tabulated, after tabulating the data is plotted to see the change in DB and WAS CPU usage.

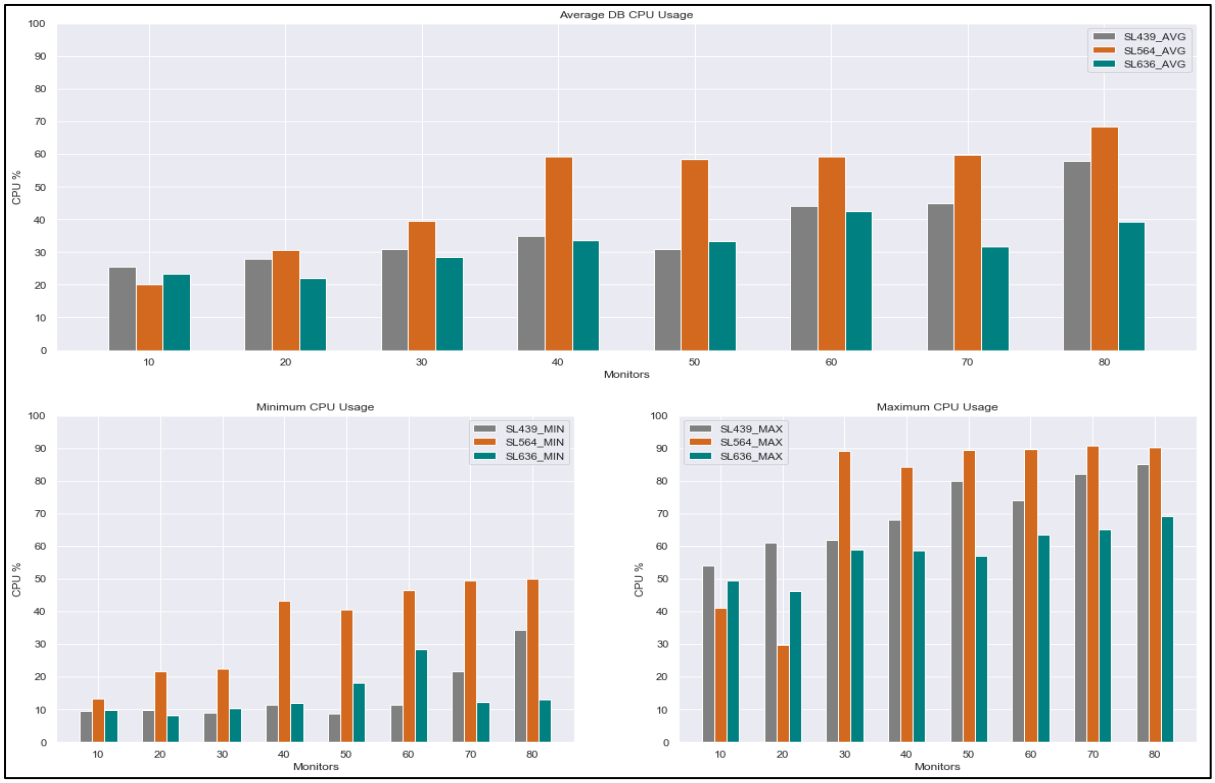

Figure 3.8 DB CPU Usage of all three configuration labs.

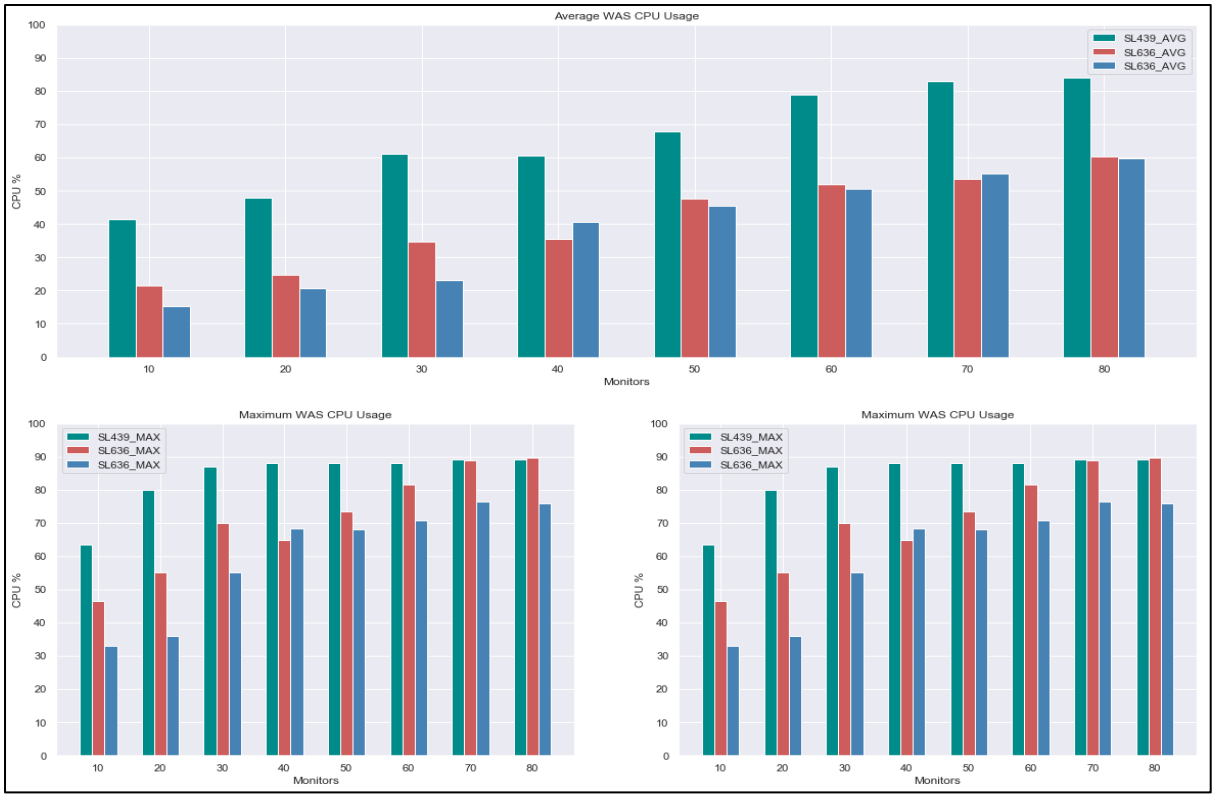

Figure 3.9: WAS CPU Usage of all three configuration labs.

From the above diagrams we can observe that,

DB CPU:

Average levels have minimal increase with increase in monitors, average CPU usage levels start from 10% and go as high as 60% when all the 80 monitors are launched, however the Large lab (SP 564) has higher DB CPU usage levels as the lab had DB aggregation of data along with the load configuration test while the other two labs were tested with only with a load configuration and without DB aggregations.

Minimum CPU levels are below 10-20% most of the time with 564 labs being an exemption, Minimum levels are mostly observed when there are no operations being performed in the monitors.

Maximum CPU levels reach 50 – 60% levels when operations are performed in the monitors until 30 monitors are launched, after 30 monitors the maximum levels keep increasing up to 80%.

WAS CPU

In case of WAS CPU levels, we can clearly see that there are high average CPU usage levels in small lab (439) and the average CPU levels keep decreasing as the configuration of the lab is upgraded.

Minimum WAS CPU levels are reach higher levels starting from 10 monitors unlike the DB CPU, even in this case we can see that the small lab has higher CPU levels compared to the Large and XXL labs.

Maximum CPU levels also exhibit the same behavior as minimum and average WAS nodes, the small and large configuration labs reach the 90% threshold levels after 30 monitors are launched.

Here is a overview of time taken when several operations were performed in a lab and the response time for the operation was captured:

	20 Monitors	**40 Monitors**	**60 Monitors**	**80 Monitors**
ACK	1	2	2	2
UNACK	1	1	1	1
Cancel	2	2	2	2
Maintenance	1	1	1	1
Resume	17	22	20-22	23
Alarm Upload	6	6	8	9
Alarm Filter	18	20	12	-

From the tabulated data we can infer that the change in response time of operations performed while launching more monitors will not be impacted except when resuming the alarms from maintenance mode.

Hence, we can infer that stepwise monitor load will impact the resources of the virtual machine while the overall user interface of the monitor will not face any issues in terms of response time due to operations performed in the monitor.

4.4 RAW AND AGGREGATED DATA ANALYSIS

When NetAct is running with a specific load configuration where counters are being sent at some measurement interval there are various metrics that will be impacted based on the both the above factors. One such KPI is the disk usage or disk space consumed per day. There are two disk usage metrics recorded for different network adaptations used in this analysis:

Network Adaptations:

- NOKMRN
- NOKGNB
- NOKLTE
- XMLNSS
- NOKRWW

Metrics

Raw disk usage

Aggregated disk usage:

- AGG TOTAL
- AGG HOUR
- AGG DAY
- AGG WEEK
- AGG BH (Busy Hour)
- AGG WEEK BH

When counters are generated Raw disk usage consumption keeps growing, the aggregation however will depend on the measurement interval, hour, day, week etc. To understand how raw and aggregation of disk space works a test was done where the load was triggered using different use cases and data was captured for the corresponding time-frames. The test was conducted with 15, 30 and 60-minute intervals with all the above adaptations integrated to the lab.

The goal of conducting the test was to:

1. Check if all adaptations use same disk usage for a specific number of counters.
2. Check if the Raw to Aggregation disk usage ratio is same for all adaptations.
3. Build a predictive model to estimate the Raw and Aggregated disk usage of a load model using machine learning.

RAW disk usage analysis:

The test was conducted on a lab (Sprint lab 398) and all adaptations were tested at different measurement intervals and load configurations and the data was collected. It was ensured that all the use cases required to estimate the raw and aggregated disk usage were tested on the lab before performing the analysis.

After performing the tests for a period of 8-10 days (2-3 days each at different measurement intervals), the data was tabulated. Using the tabulated data, estimation was done to obtain the disk usage per million, 10 million and 100 million counters using the test data.

Here is the disk usage (per million counters) observed at different measurement intervals:

Adaptation	Avg Disk per mil (MB)
NOKMRN	20.28
NOKRWW	7.638
NOKBSC	6.988
NOKLTE	7.787
XMLNSS	9.56

From the above table we can assume that the disk usage is same for NOKRWW, NOKBSC and NOKLTE but when we extrapolate the disk usage to 100 million or 10 billion counters there will be a higher delta of error values in the data. This shows that all the adaptations that were tested have different disk usage values and the predictive modelling can be conducted adaptation wise only. In order to perform predictive modelling more data was required and hence data of 5 more labs were collected to check if the average disk per million counters results of 398 lab checks out in all the 5 labs data that was collected.

The analysis resulted with very minimal error when the avg disk per million counters of 5 labs data was compared 398 lab data. Hence, data of these 5 labs along with 398 data was used to create a regression model to estimate the disk usage for any specified counters of different adaptations separately.

To test the obtained regression equations, counters data of 3 more labs were taken and the values were equated using the equations. Here are the results:

	Adaptation	Counters	Actual Disk Usage	Predicted Disk Usage	Error
petslab255	NOKMRN	1052241533	19548	21737.07854	-2189.078544
	NOKRWW	4439850048	28045.1875	28076.70881	-31.52130585
	NOKBSC	2837682000	17693.4375	17778.65099	-85.21349244
	NOKLTE	7561385016	41656	43485.89579	-1829.895794
	XMLNSS	445528699	5449	4259.254362	1189.745638
SL487	NOKMRN	1332576000	31136	29528.19412	1607.805883
	NOKRWW	2250572468	14001	14232.16261	-231.1626062
	NOKBSC	1337380802	8264.25	8378.960899	-114.7108993
	NOKLTE	2711520000	20684.875	18462.89216	2221.982836
	XMLNSS	223609441	2790.6875	2137.706256	652.981244
	NOKGNB	1700784000	13155	12115.5294	1039.470603
SL439	NOKMRN	666288000	15534	13764.09706	1769.902942
	NOKRWW	1109962512	7139.375	7019.177201	120.1977985
	NOKBSC	660014600	4159.8125	4135.124804	24.68769626
	NOKLTE	1768560000	13427	12042.2245	1384.775504
	XMLNSS	66721112	865.5	637.8538307	227.6461693
	NOKGNB	850392000	6655	6057.764699	597.2353013

Figure 3.10: Actual vs Predicted raw disk usage for test data

Initially, the error in the estimation of disk usage in the dimensioning tool was ~30 GB, with the above estimation the highest margin of error is less than 3 GB which is a better performing model than the previous equations incorporated in the dimensioning tool.

This level of accuracy was achieved only due to the tests that were conducted on 398 at a granular level and the data was collected and validated for different measurement intervals.

Raw vs Aggregated Total Ratio:

After estimating the raw disk usage, we need to analyze how raw disk usage is aggregated in different ways and the ratio between raw disk usage and aggregated disk usage among different adaptations.

The goal is to check whether the ratio of RAW data vs Aggregated data is correlating with the measurement interval at which counters are being sent w.r.t the adaptations.We start by establishing the relationship amongst different adaptations in the 5 labs data collected for raw modelling and tabulate the ratio.

The ideal aggregation ratio between Raw disk usage and Aggregated Total should be:

- 1 = 60 min counters
- 2 = 30 min counters
- 4 = 15 min counters

For example, if raw disk usage is 1 GB, the AGG Total should be approx. 256 MB if there are 15-minute counters, ~500MB if its 30 min counters and should remain 1 GB for 60 min counters.

The data was tested for all the 5 labs used for Raw modelling to obtain the relationship between Raw disk usage and AGG Total disk usage. However, the results didn’t come out as expected in the data. This was because the number of counters sent at different intervals were not known in prior, hence some labs contained only 15 min counters or only 30 min counters while some labs had counters at all intervals (15 + 30 + 60 min counters). Hence obtaining the Raw vs Aggregated Total ratio would completely depend on counter intervals which should be known in prior.

Aggregated Total vs other Aggregations:

As we know there are different aggregations that happen in NetAct in a periodic manner (AGG TOTAL, AGG HOUR, AGG DAY, AGG WEEK, AGG BH (Busy Hour), AGG WEEK BH).

Aggregated Total attributes to the disk usage that is combined of all the aggregations, i.e., AGG Total = AGG Hour + Day + BH + Week + Week BH. Hence, we need to derive the ratio between AGG Total w.r.t to all other aggregations.

Here we completely ignore the Raw disk usage and use the 5-lab data to derive the ratio between different aggregations. After obtaining the ratio values we compared if all the 5 labs had same aggregation ratio values, it turned out that AGG Day, Hour, BH had same values but Week and Week BH had change in the values. When investigated it was found out that the Week and Week BH is recorded only once in a 7-day period (Sundays) and the value keeps getting halved until the next aggregation is done. We collect the data in different days and due to halving of values in Week and Week BH, this change can be observed in the data.

After performing these calculations on 5 different labs, the ratio values for different aggregations was obtained. The ratio values were different for different aggregations and also different for different adaptations.

To check if the ratio values are accurate and efficient we obtained data 3 more labs and using only the AGG Total disk usage we computed all the other Aggregations, The test data resulted in having no or very minimal error in predicted aggregated disk usage (less than 500 MB error). when compared to actual Aggregated disk usage.

4.5 TIME SERIES ANALYSIS USING ARIMA

Time series is a sequence of data points in chronological sequence, most often gathered in regular intervals. Time series analysis can be applied to any variable that changes over time and generally speaking, usually data points that are closer together are more similar than those further apart.

Time Series Data Components

Most often, the components of time series data will include a trend, seasonality, noise or randomness, a curve, and the level. Before we go on to defining these terms, it's important to note that not all time series data will include every one of these time series components. For instance, audio files that are taken in sequence are examples of time series data, however they won't contain a seasonal component (although note they would have periodic cycles). On the other hand, most business data will likely contain seasonality such as retail sales peaking in the fourth quarter.

Here are the various components that can occur in time series data:

Level: When you read about the "level" or the "level index" of time series data, it's referring to the mean of the series.

Noise: All-time series data will have noise or randomness in the data points that aren't correlated with any explained trends. Noise is unsystematic and is short term.

Seasonality: If there are regular and predictable fluctuations in the series that are correlated with the calendar – could be quarterly, weekly, or even days of the week, then the series includes a seasonality component.

Trend: When referring to the "trend" in time series data, it means that the data has a long-term trajectory which can either be trending in the positive or negative direction. An example of a trend would be a long-term increase in a company's sales data or network usage.

Cycle: Repeating periods that are not related to the calendar. This includes business cycles such as economic downturns or expansions or salmon run cycles, but aren't related to the calendar in the weekly, monthly, or yearly sense.

When a series contains a trend, seasonality, and noise, then you can define that series by the way those components interact with each other. These interactions can be reduced to what is called either a multiplicative or additive time series.

A multiplicative time series is when the fluctuations in the time series increase over time and is dependent on the level of the series:

Multiplicative Model: Time series = t (trend) * s (seasonality) * n (noise)

An additive model is when the fluctuations in the time series stay constant over time:

Additive Model: Time series = t (trend) + s (seasonality) + n (noise)

So, an additive model's seasonality should be constant from year to year and not related to the increase or decrease in the level over time.

Knowing whether your series data is multiplicative or additive is important if you want to decompose your data into its various parts such as trend, or seasonality. Sometimes it's enough to graph your data to discover if it's an additive or multiplicative series. But, when it's not, you can decompose your data and compare the ACF values to discover the correlation between data points when testing both an additive model and a multiplicative model.

Decomposition

Decomposition is the deconstruction of the series data into its various components: trend, cycle, noise, and seasonality when those exist. Two different types of classic decomposition include multiplicative and additive decomposition.

The purpose of decomposition is to isolate the various components so you can view them each individually and perform analysis or forecasting without the influence of noise or seasonality.

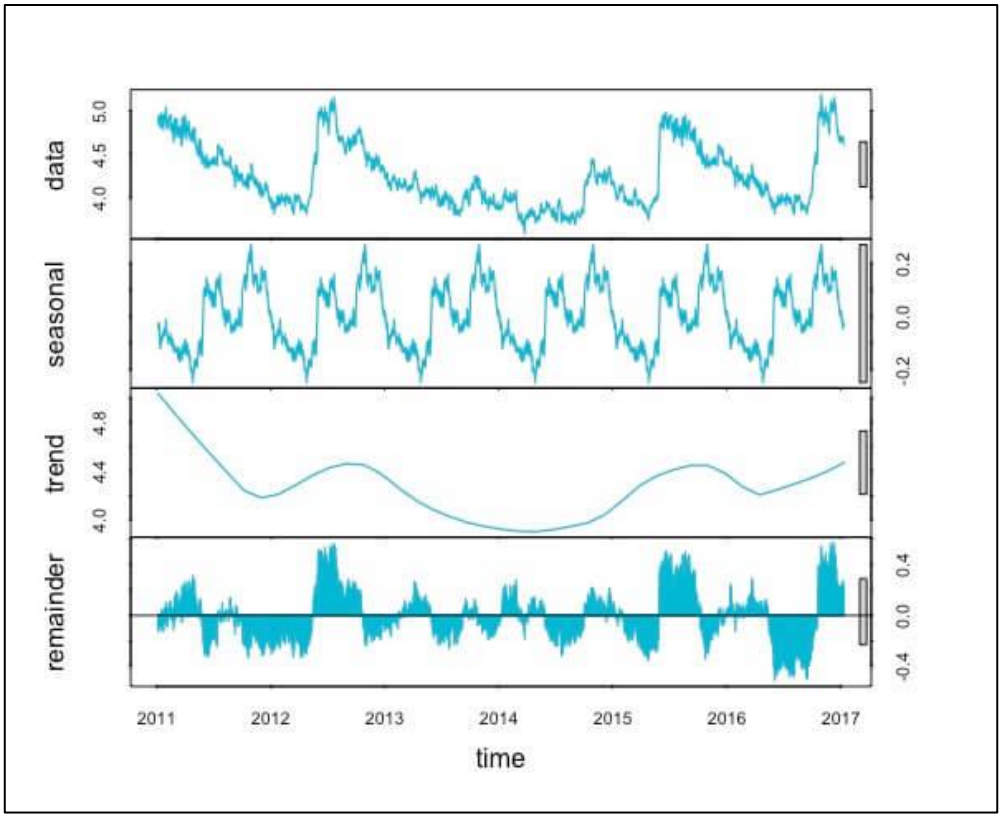

Figure 3.11: Seasonal decomposition

While it was difficult to initially see the trend in the original observed data, once our series is decomposed into its various parts, it's much easier to see the seasonality, trend, and noise or randomness in our data. To get seasonally adjusted data, you can subtract the seasonal component from the decomposed data.

Moving Averages

We already talked about how there is randomness, or noise in our data, and how to separate it with decomposition, but sometimes we simply want to reduce the noise in our data and smooth out the fluctuations from the noise in order to better forecast future data points.

A moving average model leverages the average of the data points that exist in a specific overlapping subsection of the series. An average is taken from the first subset of the data, and then it is moved forward to the next data point while dropping out the initial data point. A moving average can give you information about the current trends, and reduce the amount of noise in your data. Often, it is a preprocessing step for forecasting.

Time Series Forecasting Models

Forecasting is one of the most relevant tasks when working with time series data, but it's hard to know where to get started. Autoregressive Integrated Moving Average is popular for fairly accurate and quick forecasting of time series.

Autoregressive Integrated Moving Average:

The Autoregressive Integrated Moving Average, or ARIMA model, is a univariate linear function that is used for predicting future data points based on past data. ARIMA combines the models own past data points to determine future points versus a linear regression model that would rely on an independent variable to predict the dependent variable such as using treasury note rates to predict mortgage rates. Because of ARIMA's reliance on its own past data, a longer series is preferable to get more accurate results.

ARIMA relies on a stationary model, meaning that the mean, variance and other measures of central tendency must remain the same over the course of the series. To achieve a stationary model, you would need to remove trend, seasonality, and business cycle if that exists. The differencing method is used to fit the model by subtracting the current value from the previous data point, including the lag. Note that the lag is simply the difference between values from the same period.

For instance, if you have data that has monthly seasonality, you would take each data point, subtract it from the previous data point in the same month from the previous year, and so on in order to difference your data and create a stationary model.

In an ARIMA model you'll find that you can adjust various parameters which are:

- **p (AR**) which is the number of autoregressions,
- **d (I)** which is the number of differences, and
- **q (MA)** which is the forecast error coefficient, which together, make up (AR, I, MA).

Many statistical packages allow you to either set the ARIMA parameters manually or with an automated function that best fits your model. If you choose to select your parameters manually, you can estimate the appropriate ones with an autocorrelation and partial autocorrelation function (ACF, PCF).

In NetAct, there are several virtual machines where the time series data is continuously generated for days, weeks and months, but most of the tests don't exhibit the trend, seasonality or pattern in the data that is being generated.

A 14-day stability test is conducted once in few months before a new build is released to a customer, this test is conducted to ensure that there are no bugs, blockers or issues in the build that's being released. During this test the load configuration and the number of counters being sent is fixed for all the days, we can say that the pattern and behavior of metrics is similar in all the recorded days.

The goal of this chapter to build a time series model where the objective is to predict the behavior of the Database (DB) virtual machine during this 14-day stability test. The data is captured from Grafana and the metric being used is the DB CPU.

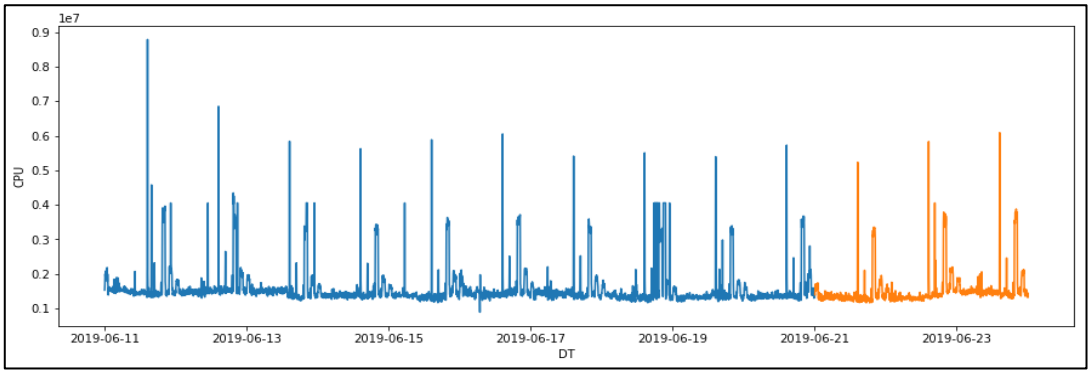

Fig 3.12: 14-day Stability data

The above diagram shows the DB behavior for 14-day period, 10-day data of the CPU usage is considered as train data and the remaining 4 days is used for testing the model.

The first process in ARIMA is to check whether the data is stationary, ad-fuller test was done to check if the data exhibits stationary behavior and the resultant p-value was greater tha 0.05 hence data needs to be made stationary, differencing of data is an approach to make the data stationary, data can be differenced once or twice and can be visually checked using rolling mean and standard deviation.

After making the data stationary, we run the ad-fuller test again to check and validate our data. A seasonal decomposition plot contains 4 subplots (data, seasonality, trend and residuals) through which we can see the properties of data before building the ARIMA model.

However, In our data the seasonality was not observed which is a crucial factor for the ARIMA model to forecaset the data,

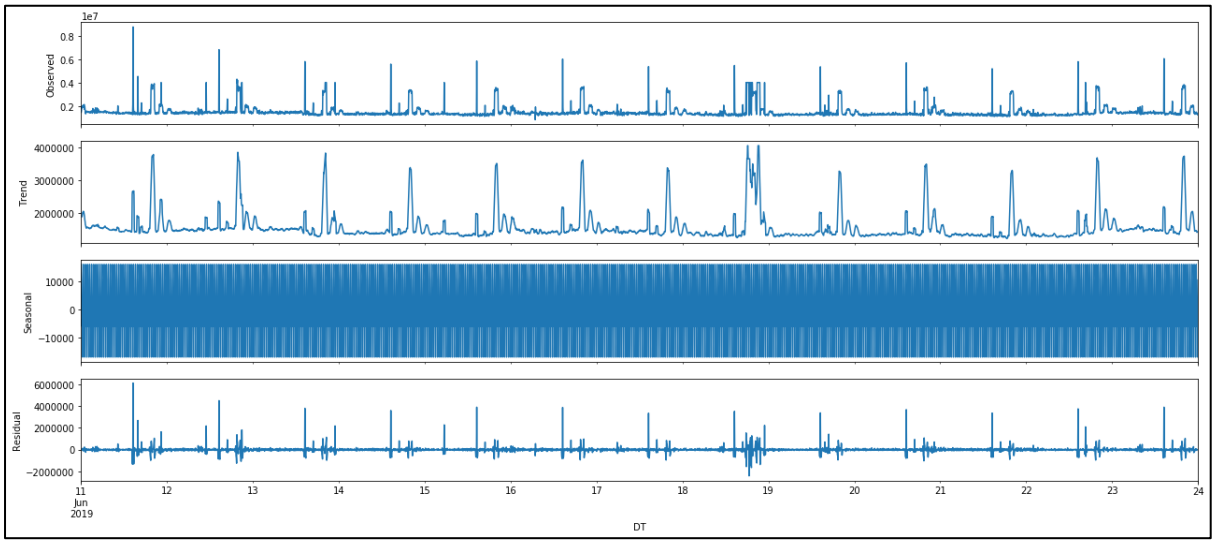

Figure 3.13: Seasonal decomposition in 14-day stability data

To obtain the seasonality various approaches like SARIMA (Seasonal ARIMA), Auto ARIMA, normalizing the data and obtaining different stability run datasets, performing analysis was done, but the end result remained the same.

When the model was built without seasonality the data using ACF and PACF plots with different p, d and q values the result always remained the same, The model would well for the test data but when forecasted it would result in straight line i.e., It would take the mean value of the test data and plot the mean value as forecasted data.

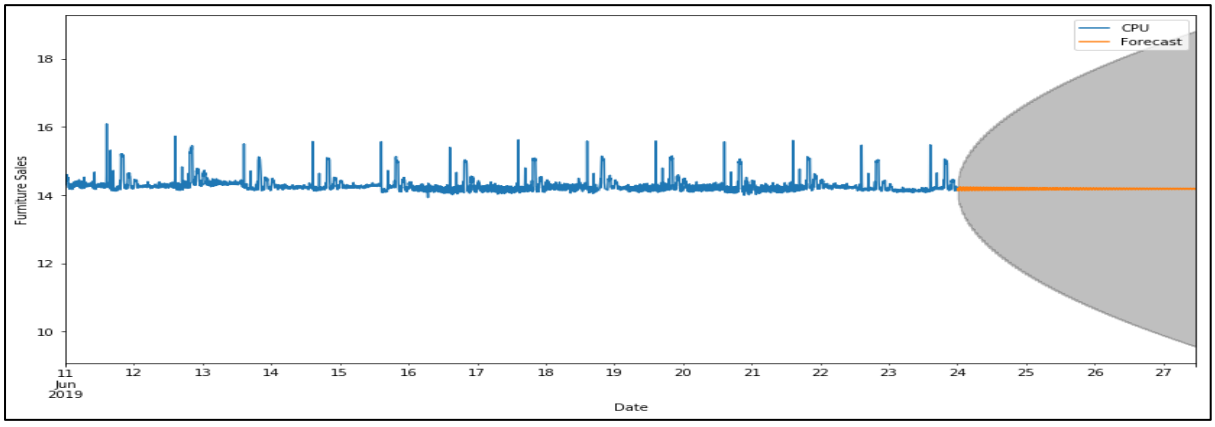

Figure 3.14: Forecasted data of ARIMA model

Hence, the stability run test behavior couldn't be predicted using ARIMA due to absence of seasonality in the data.

CHAPTER 4

CONCLUSIONS AND FUTURE SCOPE OF WORK

CONCLUSIONS

Obtaining inferences and tabulated data by performing exploratory and explanatory data analysis, statistical analysis and machine learning will help in achieving accurate and concise results.

Performing these analysis and testing methods at the granular level will help the users to understand the NetAct methodology in a better manner which in turn helps in making better decisions and improvements while deploying to the customers.

The tabulated data obtained from various use-cases is integrated to the NetAct Dimensioning Tool where previous data is replaced with the tabulated data obtained by performing predictive analysis.

NetAct Dimensioning Tool is used for various greenfield and deployment scenarios where the tabulated data of various use cases and KPIs is highly significant in order to provide infrastructure for the buying customers, by obtaining the data using predictive analysis and machine learning methods both NetAct vendors and customers will be beneficial.

FUTURE SCOPE

NetAct is highly sophisticated high-performance tool which generates huge amount of data in daily basis, performing similar analysis on different use cases and metrics by doing exploratory and explanatory data analysis can help in obtaining relationship between one or more metrics.

We can integrate multivariate analysis into Grafana and Zabbix to help the NetAct vendors and users where dependence and relationship between multiple metrics can be understood in the ongoing test cases on different load configurations. Once multivariate analysis data is integrated to network monitoring tools such as Grafana and Zabbix, we can use machine learning methods, statistical techniques and predictive modelling practices to improve NetAct simulators and customer labs working methodologies.

CHAPTER 5

TOOLS AND LIBRARIES USED

Programming Languages: Python, R, SQL

IDE: Jupyter (Anaconda), R Studio

Libraries: Sklearn, Pandas, Numpy, Seaborn, Scikit-learn

Machine Learning Algorithms: Linear Regression, AR, MA and ARIMA

Network monitoring tools: Grafana, Zabbix

CHAPTER 6

BIBLIOGRAPHY

ARIMA and Time Series:

https://medium.com/swlh/an-introduction-to-time-series-analysis-ef1a9200717a

https://towardsdatascience.com/analyzing-time-series-data-in-pandas

https://www.machinelearningplus.com/time-series/time-series-analysis-python

NetAct Documentation

http://belk.netact.noklab.net/ (Intranet only)

Nokia, “NetAct Virtualized OSS that goes beyond network management”,

Flextronics Software System, "FCAPS", White Paper.

Nokia Siemens & Networks, “OSS architecture –the next generation”.

Grafana

https://grafana.com/docs/

https://github.com/grafana/grafana ▪

Zabbix

https://www.zabbix.com/documentation/2.4/manual

https://github.com/zabbix/zabbix

https://alexanderzobnin.github.io/grafana-zabbix/