SYSTEM MONITORING AND INSTRUMENTATION IN VIRTUAL

A PROJECT REPORT

Submitted by

RITIKAA R [211420104228]

SRIVARSHINI T [211420104269]

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATION

Certified that this project report "**SYSTEM MONITORING AND INSTRUMENTATION IN VIRTUAL"** is the bonafide work of "**RITIKAA R [21142104228]** AND **SRIVARSHINI T [211420104269]**" who carried out the project work under my supervision.

Signature of the HOD with date	**Signature of the Supervisor with date**
Dr.L.JABASHEELA M.E.,Ph.D.,	**Dr.A.HEMLATHADHEVI Ph.D.,**
Professor and Head, Department of Computer Science and Engineering, Panimalar Engineering College, Chennai- 123	Associate Professor, Department of Computer Science and Engineering, Panimalar Engineering College, Chennai- 123

Submitted for the Project Viva – Voce examination held on __________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **RITIKAA R [211420104228]** and **SRIVARSHINI T [211420104269]** hereby declare that this project report titled **“SYSTEM MONITORING AND INSTRUMENTATION IN VIRTUAL”**, under the guidance of **Dr.A.HEMLATHADHEVI M.E.,Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any other degree in any university by us.

RITIKAA R [211420104228]

SRIVARSHINI T [211420104269]

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt. C. VIJAYARAJESWARI , Dr. C. SAKTHIKUMAR , M.E. , Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.** for his timelyconcern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department , **Dr. L .JABASHEELA, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank our Project Coordinator **Dr.K.VALARMATHI, M.E., Ph.D.,** and our Project Guide **Dr.A.HEMLATHADHEVI, M.E., Ph.D.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

Our sincere thanks to **Mr.E.MAHENDRAN, SCIENTIST-E, C-DAC, CHENNAI**, for this kind support. Finally, I would like to thank all those who were directly or indirectly helpful in carrying out this research.

RITIKAA R [211420104228]

SRIVARSHINI T [211420104269]

ABSTRACT

The abstract provides a comprehensive overview of the pivotal role that monitoring plays in contemporary system management. It emphasizes how monitoring offers real-time insights into system behaviour, performance metrics, and potential issues, ensuring the efficient allocation and utilization of resources for optimized operational efficiency. By strategically deploying instrumentation to monitor CPU, memory, and network usage, organizations can effectively manage their systems. Integral to this process is the thorough analysis of logs, traces, and metrics, which collectively provide a multifaceted understanding of system behaviour. Logs serve as meticulous event records, crucial for diagnosing issues and conducting system audits, while traces illuminate operational flows, shedding light on error propagation pathways. Additionally, metrics offer quantitative data on crucial performance indicators such as throughput, response time, and error rates, empowering stakeholders to make informed decisions about resource allocation and capacity planning. The abstract also highlights the significance of monitoring within the burgeoning realm of Internet of Things (IoT) systems, which are vital for managing expansive networks of interconnected devices. Effective monitoring solutions, tailored to diverse application domains, necessitate meticulous categorization of monitoring requirements based on specific use cases and system characteristics. In conclusion, the abstract underscores the critical role of monitoring as a cornerstone of contemporary system management practices. It emphasizes how monitoring facilitates real-time insights, optimizes resource allocation, and ensures overall system health and efficiency through the analysis of logs, traces, and metrics. By embracing effective monitoring practices, organizations can enhance their operational efficiency, improve system reliability, and stay ahead in today's dynamic technological landscape.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Internet-of-Things (IoT) systems differ greatly in terms of their design, complexity, attack surface, geographic dispersion, heterogeneity, and other factors because of their vast range of applications. IoT systems differ from "traditional" distributed applications because of this heterogeneity. These days, cloud computing principles are typically followed when operating the latter. Elements like the DevOps pattern and agile principles have gained popularity for managing the development and operation of cloud applications .Similar to this, modern cloud services are highly automated, which lowers the possibility of human error while enabling quick responses to outtakes, overload, and other context changes.

The majority of IoT systems are built with cloud-centric core services that function similarly to conventional cloud apps. However, there is an open research challenge to enable a comprehensive administration of IoT systems based on cloud techniques, which requires incorporating the expanding opportunities provided by fog and edge computing.

Furthermore, in order to regard the cloud-fog-edge continuum as one massive cloud-like infrastructure, standard cloud computing features like automation, scalability, elasticity, and service models must be pushed into the Internet of Things (IoT) domain. Autonomous computing often uses the well-known MAPE loop (Monitor, Analyze, Plan, and Execute) paradigm with various modifications and potential Knowledge/Machine Learning extensions (MAPE-K) to enable a high degree of automation. Understanding the system's past, present, and future states is the foundation for all wise decisions made in MAPE cycles.

1.2 PROBLEM DEFINITON

Due to the diverse range of interconnected devices and the lack of standardized monitoring protocols in IoT environments, traditional system monitoring solutions face significant difficulties in providing comprehensive insights into system health and performance.

The dynamic nature of IoT ecosystems, characterized by continuous data streams and rapidly changing device configurations, further exacerbates the complexity of system monitoring, necessitating adaptive monitoring solutions capable of real-time analysis and response.

The distributed nature of IoT deployments, with devices spread across geographically diverse locations and often operating in challenging environments, introduces additional hurdles for effective system monitoring, requiring robust monitoring solutions capable of remote management and fault detection.

1.3 LITERATURE REVIEW

INTRODUCTION

A literature review serves as a comprehensive examination of existing knowledge and methodological approaches pertaining to a specific subject. It synthesizes secondary sources, analyzing published information within a defined topic or timeframe. Its primary objective is to provide readers with a thorough understanding of the current state of research in a given field, laying the groundwork for further investigation or future research endeavors. Often preceding a research proposal, a literature review goes beyond a simple summary of sources, delving into critical points, identifying gaps in knowledge, and offering insights that inform subsequent research directions.

Medical device development projects must follow proper directives and regulations to be able to market and sell the end-product in their respective territories. The regulations describe requirements that seem to be opposite to efficient software development and short time-to-market. As agile approaches, like DevOps, are becoming more and more popular in software industry, a discrepancy between these modern methods and traditional regulated development has been reported. Although examples of successful adoption in this context exist, the research is sparse. The goal of this study is twofold: to analyse the present state of DevOps adoption in the regulated medical device environment, and to offer a checklist based on that evaluation for implementing DevOps in that setting.[1]

Although academic and business professionals are now advocating for the transition from big, centralized Cloud Computing infrastructures to smaller, massively dispersed ones at the network's edge, management mechanisms to operate and use such infrastructures are still lacking. In this paper, we focus on the monitoring service, which is an essential component of any management system in charge of running a distributed infrastructure. Several solutions have previously been offered for cluster, grid, and cloud systems. However, none are particularly applicable in the Fog/Edge scenario. Our study aims to pave the road for a comprehensive monitoring solution for a Fog/Edge infrastructure that hosts next-generation digital services.[2]

Internet-of-Things (IoT) ecosystems tend to grow in size and complexity, since they include a wide range of heterogeneous devices that span several architectural IoT levels (e.g., cloud, edge, sensors). Furthermore, as IoT systems become more integrated into critical infrastructures, they place a greater emphasis on service resilience. This leads to a wide range of research projects aimed at improving the resilience of these systems. In this article, we present a systematic overview of existing scientific attempts to make IoT systems durable. In specifically, we first examine the taxonomy and classification of resilience and resilience mechanisms, followed by a study of cutting-edge resilience mechanisms proposed by research and applicable to the Internet of Things.[3]

Managing the data provided by Internet of Things (IoT) sensors is one of the most difficult tasks when installing an IoT system. Traditional cloud-based IoT systems are hampered by the vast scale, heterogeneity, and excessive latency seen in some cloud ecosystems. A distributed and federated compute model can help decentralize applications, management, and data analytics within the network. This approach is known as fog computing. This document defines fog and mist computing and how it relates to cloud-based IoT paradigms. This document describes key components of fog computing, such as service models and deployment methodologies. It also provides a foundation for understanding fog computing and its potential applications.[4]

Recently, a promising movement has emerged from centralized compute to decentralized edge computing near end users to provide cloud applications. A complete monitoring approach is required to assure the quality of service (QoS) of such applications as well as the quality of experience (QoE) for end users. Requirement analysis is a critical software engineering effort throughout the application lifecycle; however, the requirements for monitoring systems in edge computing scenarios are not yet fully defined.[5]

The Internet of Things (IoT) aims to establish an interconnected ecosystem that allows gadgets to communicate via the Internet. To achieve this purpose, the ecosystem's Device to Device (D2D) communication technologies must work together effectively. Currently, these technologies function in vertical silos using various protocols. We investigate the issues involved with the integration and interoperability of different D2D technologies, focusing on network layer functions such as addressing, routing, mobility, security, and resource optimization. We describe the limitations of the current TCP/IP architecture for D2D communication in the IoT environment. We also go over some of the constraints of the 6LoWPAN design and explain how it has been used for D2D communication.[6]

Cloud infrastructures can allow resource sharing for multiple applications and often meet the needs of the majority of them. However, in order to make the best use of these resources, automatic orchestration is essential. Automatic orchestration systems are typically predicated on the observability of the infrastructure, although this is insufficient in some instances. Certain types of applications have special needs that are difficult to achieve, such as low latency, high bandwidth, and high computing power. To achieve these needs, orchestration must be built on multilevel observability, which entails gathering data at both the application and infrastructure levels. [7]

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISITING SYSTEM

Traditional monitoring methods are ill-equipped to handle the dynamic nature and scalability demands of modern IoT environments, leading to limited visibility, reactive problem-solving and inefficient resource allocation.

The reliance on static configurations and blanket monitoring strategies in existing systems hinders adaptability to the dynamic changes within IoT environments, leading to missed opportunities for proactive problem resolution and optimization of resource allocation.

2.2 PROPOSED SYSTEM

The primary objective of this research is to design, develop, and evaluate a novel monitoring system for cloud-to-thing applications, leveraging pinpoint configuration and predictive analytics to enhance visibility, reliability, and scalability.

It also involves integrating Pinpoint, a cloud-based application monitoring tool, to automate the log extraction process. Pinpoint can collect logs and metrics in real-time from the cloud-based application's various components.

With Pinpoint in place, the system gains real-time monitoring capabilities. It can immediately detect anomalies, errors, or performance issues, allowing for proactive intervention and minimizing potential downtimes.

2.3 IMPLEMENTATION ENVIRONMENT

2.3.1 SOFTWARE IMPLEMENTATION

- Ubuntu LS-20.0.7
- Java Development Kit-jdk 21 version
- Spring Boot initializer
- Idea Community IDE
- Hbase 2.5.7
- Wave front
- Pinpoint Agent 2.5.3

2.3.2 HARDWARE IMPLEMENTATION

- Processor: Intel i5 or above
- Memory (RAM): 16 GB
- Hard Drive: 32 GB
- Internet Connection

CHAPTER 3
SYSTEM DESIGN

3.1 UML DIAGRAM

ACTIVITY DIAGRAM

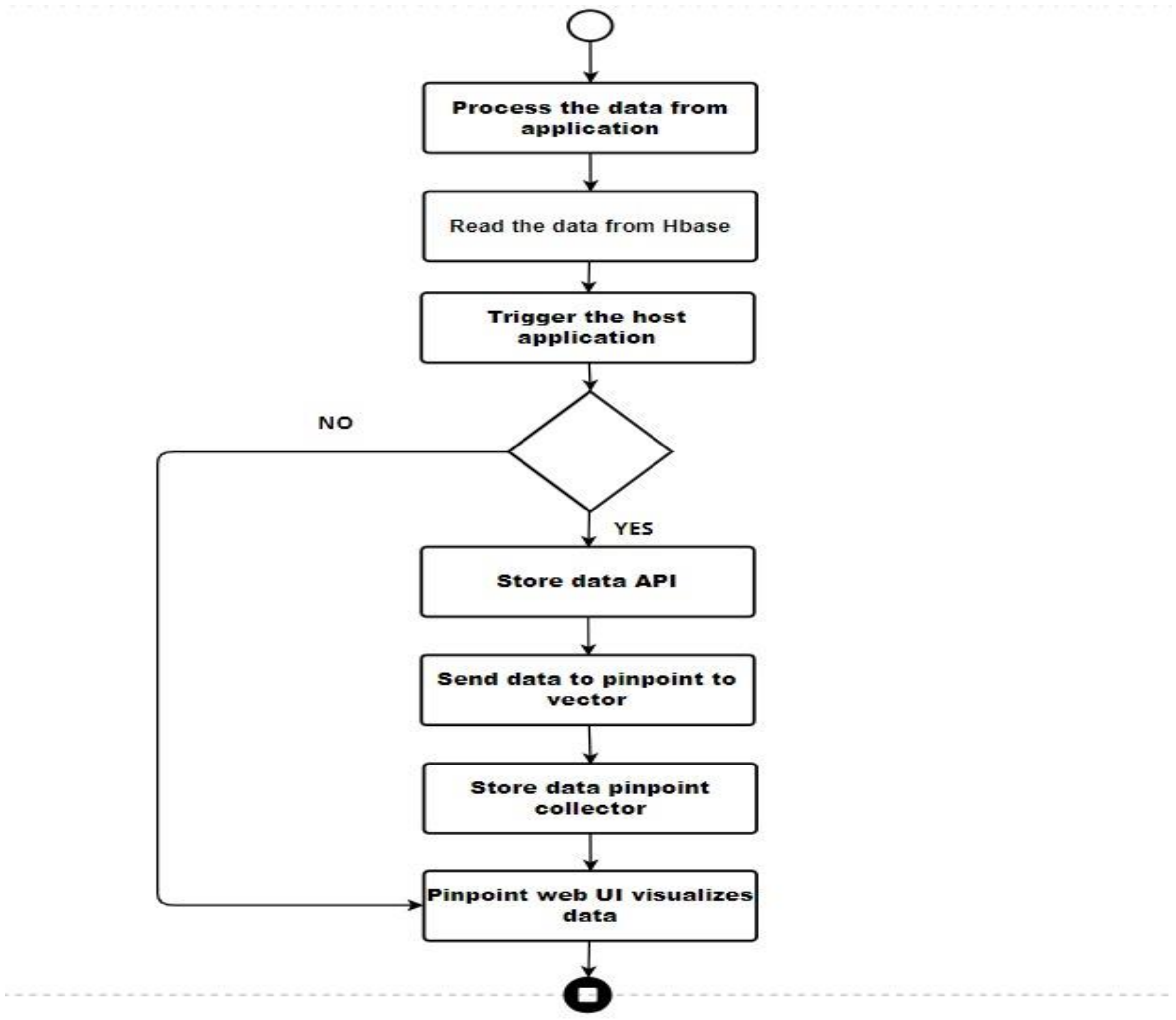

Process the data from application
Read the data from Hbase
Trigger the host application
NO
YES
Store data API
Send data to pinpoint to vector
Store data pinpoint collector
Pinpoint web UI visualizes data

Fig 3.1.1 ACTIVITY DIAGRAM

The activity diagram illustrates Application 1 triggering "Process the Data," reading from HBase, and sending data to Pinpoint Agent. Pinpoint Agent receives, processes, and forwards data to Pinpoint Collector, which stores it. Pinpoint Web UI then visualizes the stored data.

DEPLOYEMENT DIAGRAM

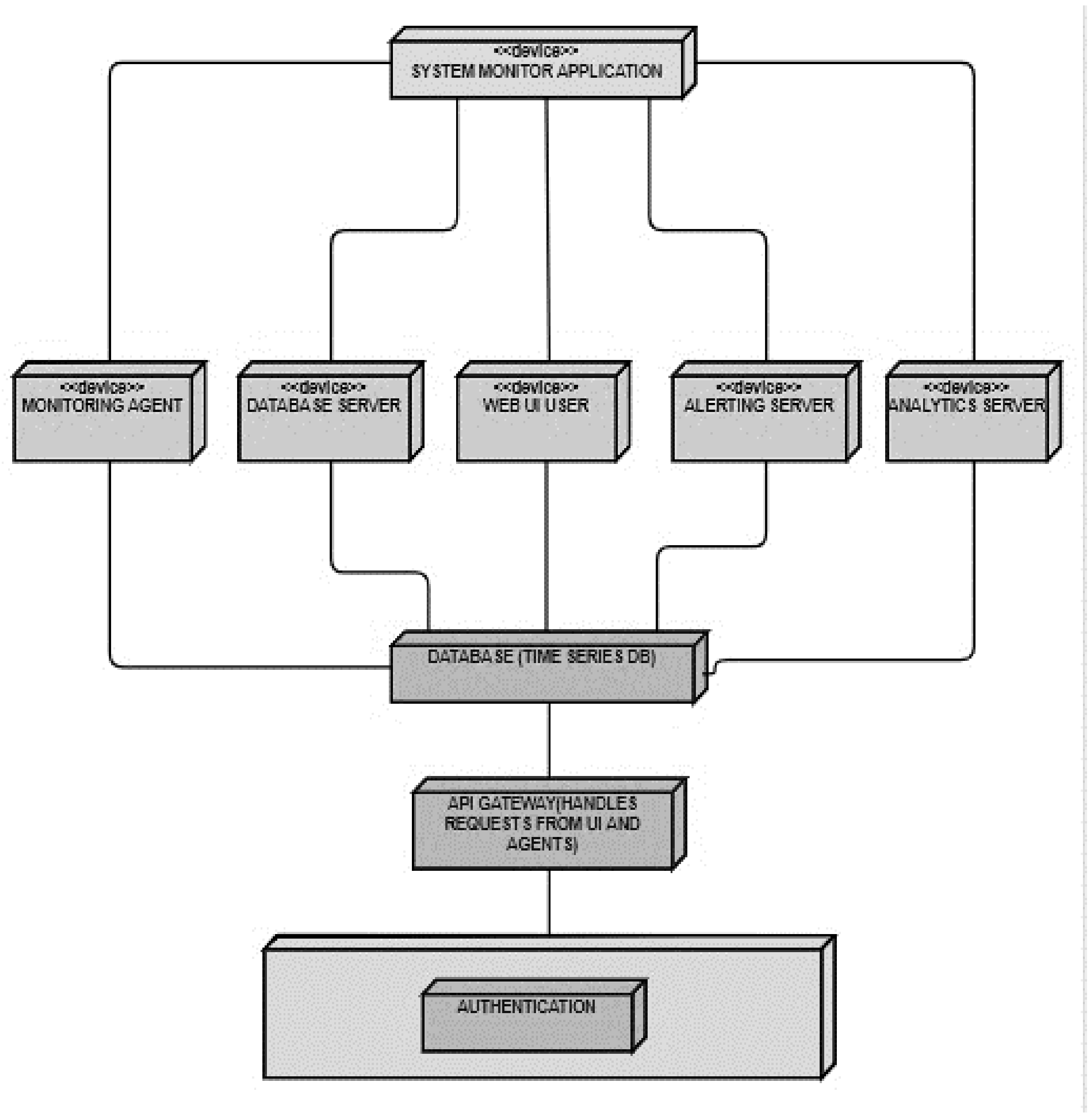


Fig 3.1.2 DEPLOYMENT DIAGRAM

Deployment diagram: Application Server hosts App 1 and Pinpoint Agent; Pinpoint Collector and Web UI on separate servers. HBase stores performance data.

CHAPTER 4
SYSTEM ARCHITECTURE

4.1 ARCHITECTURE DESIGN

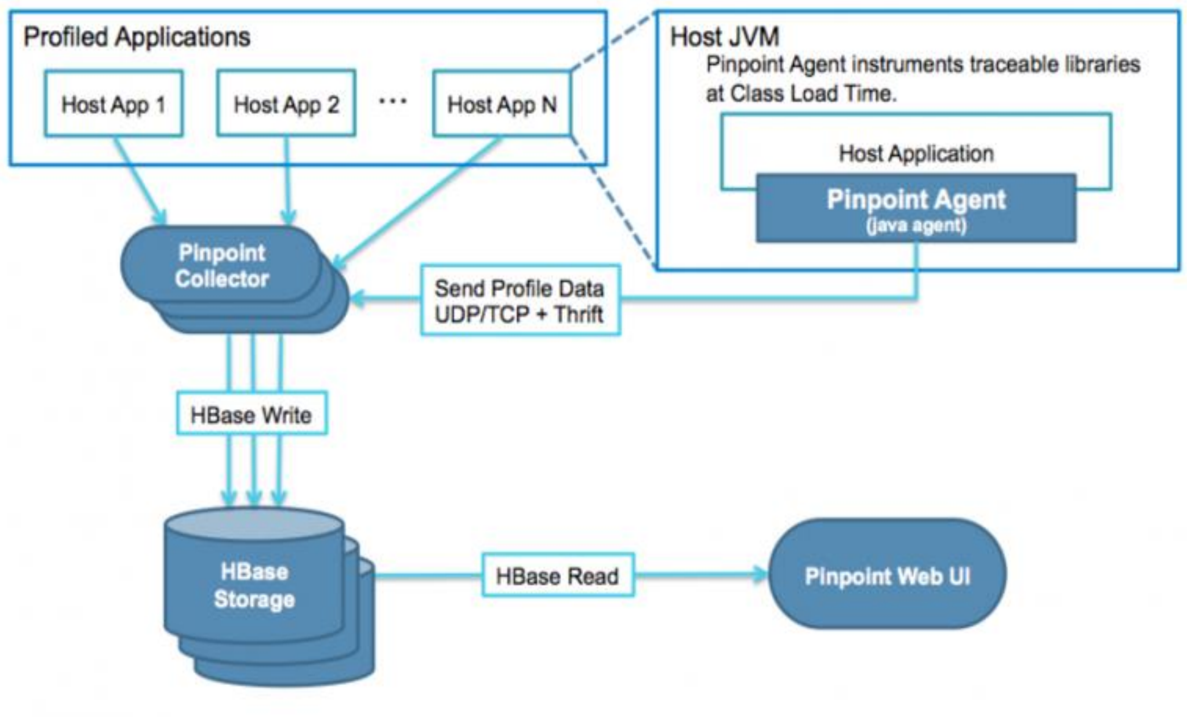


Fig 4.1.1 Architecture diagram

HOST APPLICATION

These are the Java applications that you want to monitor. They can be deployed on one or more servers**.**

HBASE:

This is a NoSQL database that stores the performance data collected by the Pinpoint collector.Pinpoint uses HBase as its storage backend for the Collector and the Web.Tables in HBase are automatically shared into regions, and each region is assigned to a region server. This sharing mechanism allows HBase to **efficiently distribute** data across the cluster and balance the load among region servers.To set up your own cluster, take a look at the **HBase website** for instructions. Once you have HBase up and running, make sure the Collector and the Web are configured properly and are able to connect to HBase.

PINPOINT COLLECTOR

This is a central server that collects the performance data from the Pinpoint agents. It aggregates the data and stores it in a database. Set up the Pinpoint Collector, which aggregates performance data from application agents. Pinpoint Collector is a crucial component in the Pinpoint application performance monitoring (APM) system. It serves as the **central point for aggregating, storing, and managing performance** data collected from various application instances by Pinpoint agents. The collector plays a key role in providing a centralized repository for performance metrics, allowing users to analyze and visualize the data through the Pinpoint Web UI.

PINPOINT AGENT

This is a Java agent that is installed on each host application. It collects performance data about the application, such as CPU usage, memory consumption, and response times. It provides an intuitive and interactive interface for monitoring and managing the performance of Java applications in real-time and over historical periods.

PINPOINT WEB

This is a web-based user interface that allows you to view the performance data collected by the Pinpoint system.

4.2 MODULE DESIGN SPECIFICATION

HBASE TABLE CREATION:

Start Loop: Begin an infinite loop to continuously monitor the HBase cluster.

Get Region Servers: Obtain a list of all region servers in the HBase cluster using the getHBaseRegionServers() function.

Iterate Through Region Servers: For each region server in the list obtained:

a. Check Status: Use the check HBaseRegionServerStatus(regionserver) function to determine the status of the region server.

b. Evaluate Status: If the status is "healthy": Log a message indicating that the region server is healthy.

If the status is "unhealthy": Log a message indicating that the region server is unhealthy and attempt to restart it using the restart HBaseRegionServer(regionserver) function.

Log a message confirming that the region server has been restarted.

If the status is neither "healthy" nor "unhealthy": Log a message indicating an unknown status for the region server.

Sleep: Pause the execution for 5 minutes (300 seconds) using the sleep(300) function before checking the cluster again.

End Loop: Repeat the process indefinitely.

This algorithmic process continuously checks the status of each region server in the HBase cluster, takes appropriate actions based on the status, and repeats the process at regular intervals to ensure ongoing monitoring and maintenance of the cluster.

PINPOINT COLLECTOR

Set Collector Address and Port: Assign the collector address as "collector.example.com".

Set the collector port as 9090.

Configure Collector Address and Port: Call the function "configureCollectorAddress()" with the collector address and port as arguments to configure the Pinpoint collector with the provided address and port.

Start Collector: Invoke the function "startCollector()" to begin the Pinpoint collector.

Log Collector Start: Generate a log message indicating that the Pinpoint collector has started and is listening on the configured address and port.

This procedure ensures that the Pinpoint collector is properly configured with the specified address and port, started, and ready to listen for incoming data.

PINPOINT AGENT

Set Agent ID: Define the agent ID as "agent-1".

Set Collector Address and Port: Assign the collector address as "collector.example.com".

Set the collector port as 9090.

Set Application Name: Define the application name as "MyApplication".

Configure Agent ID: Call the function "configureAgentID()" with the agent ID as an argument to set the ID for the Pinpoint agent.

Configure Collector Address and Port: Call the function "configureCollectorAddress()" with the collector address and port as arguments to configure the Pinpoint agent with the provided address and port.

Configure Application Name: Call the function "configureApplicationName()" with the application name as an argument to set the name for the Pinpoint agent's application.

Start Agent: Invoke the function "startAgent()" to initiate the Pinpoint agent.

Log Agent Start: Generate a log message indicating that the Pinpoint agent has started with the specified ID and application name.

This procedure ensures that the Pinpoint agent is properly configured with the specified ID, collector address, port, and application name, and is ready to start monitoring the designated application.

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1 BACKEND CODING

MODULE 1 PINPOINT COLLECTOR

Hbase.xml

```
<configuration>

  <property>

    <name>hbase.rootdir</name>

    <value>file:///home/pinpoint/hbase</value>

  </property>

  <property>

    <name>hbase.zookeeper.property.dataDir</name>

    <value>/home/pinpoint/zookeeper</value>

  </property>

  <property>

    <name>hbase.master.port</name>

    <value>60000</value>

  </property>

  <property>

    <name>hbase.regionserver.port</name>
```

```
        <value>60020</value>
    </property>
</configuration>
```

Pinpoint Collector.java

```
package com.navercorp.pinpoint.collector;
import com.navercorp.pinpoint.collector.config.ClusterModule;
import com.navercorp.pinpoint.collector.config.CollectorConfiguration;
import com.navercorp.pinpoint.collector.config.CollectorProperties;
import com.navercorp.pinpoint.collector.config.FlinkContextModule;
import com.navercorp.pinpoint.collector.config.MetricConfiguration;
import com.navercorp.pinpoint.collector.grpc.ssl.GrpcSslModule;
import com.navercorp.pinpoint.common.server.CommonsServerConfiguration;
import com.navercorp.pinpoint.realtime.collector.RealtimeCollectorModule;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Import;
import org.springframework.context.annotation.ImportResource;
@Configuration
@ImportResource({
        "classpath:applicationContext-collector.xml",
        "classpath:servlet-context-collector.xml",
})
@Import({
        CollectorAppPropertySources.class,
        CommonsServerConfiguration.class,
        TypeLoaderConfiguration.class,
        FlinkContextModule.class,
        CollectorConfiguration.class,
        CollectorHbaseModule.class,
        CollectorGrpcConfiguration.class,
        ClusterModule.class,
        MetricConfiguration.class,
        GrpcSslModule.class,
        RealtimeCollectorModule.class,
})
@ComponentScan(basePackages = {
        "com.navercorp.pinpoint.collector.handler",
        "com.navercorp.pinpoint.collector.manage",
        "com.navercorp.pinpoint.collector.mapper",
        "com.navercorp.pinpoint.collector.util",
        "com.navercorp.pinpoint.collector.service",
})
public class PinpointCollectorModule {
    @Bean

    public CollectorProperties collectorProperties() {

        return new CollectorProperties();}}
```

MODULE 2 : PINPOINT WEB

SERVER APPLICATION:

```
package server;
import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.util.Assert;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;
import java.util.Map;
import java.util.Set;
@Slf4j
@SpringBootApplication
public class ServiceApplication {
    public static void main(String[] args) {
        log.info("starting server");
        SpringApplication.run(ServiceApplication.class, args);
    }
}
@RestController
class AvailabilityController {
    private boolean validate(String console) {
        return StringUtils.hasText(console) &&
                Set.of("ps5", "ps4", "switch", "xbox").contains(console);
    }
    @GetMapping("/availability/{console}")
    Map<String, Object> getAvailability(@PathVariable String console) {
        return Map.of("console", console,
                "available", checkAvailability(console));
    }
    private boolean checkAvailability(String console) {
        Assert.state(validate(console), () -> "the console specified, " + console + ",
is not valid.");
        return switch (console) {
            case "ps5" -> throw new RuntimeException("Service exception");
            case "xbox" -> true;
            default -> false;
        };}}
```

CLIENT APPLICATION:

```
package client;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.event.ApplicationReadyEvent;
import org.springframework.context.ApplicationListener;
import org.springframework.context.annotation.Bean;
import org.springframework.stereotype.Component;
import org.springframework.web.reactive.function.client.WebClient;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;
import java.time.Duration;
@Slf4j
@SpringBootApplication
public class ClientApplication {
  public static void main(String[] args) {
    log.info("starting client");
    SpringApplication.run(ClientApplication.class, args);}
  @Bean
  WebClient webClient(WebClient.Builder builder) {
    return builder.build();}
  @Bean
  ApplicationListener<ApplicationReadyEvent>          ready(AvailabilityClient
client) {
    return applicationReadyEvent -> {
      for (var console : "ps5,xbox,ps4,switch".split(",")) {
        Flux.range(0, 20).delayElements(Duration.ofMillis(100)).subscribe(i -
            client
                .checkAvailability(console)
                .subscribe(availability ->
                    log.info("console:  {},  availability:  {}  ",  console,
availability.isAvailable()))); }};}} @Data
@AllArgsConstructor
@NoArgsConstructor
class Availability {
  private boolean available;
  private String console;}
```

```
@Component
@RequiredArgsConstructor
class AvailabilityClient {
   private final WebClient webClient;
   private static final String URI = "http://localhost:8083/availability/{console}";
   Mono<Availability> checkAvailability(String console) {
      return this.webClient
            .get()
            .uri(URI, console)
            .retrieve()
            .bodyToMono(Availability.class)
            .onErrorReturn(new Availability(false, console));
   }
}
```

CHAPTER 6

PERFORMANCE ANALYSIS

6.1 PERFORMANCE PARAMETERS

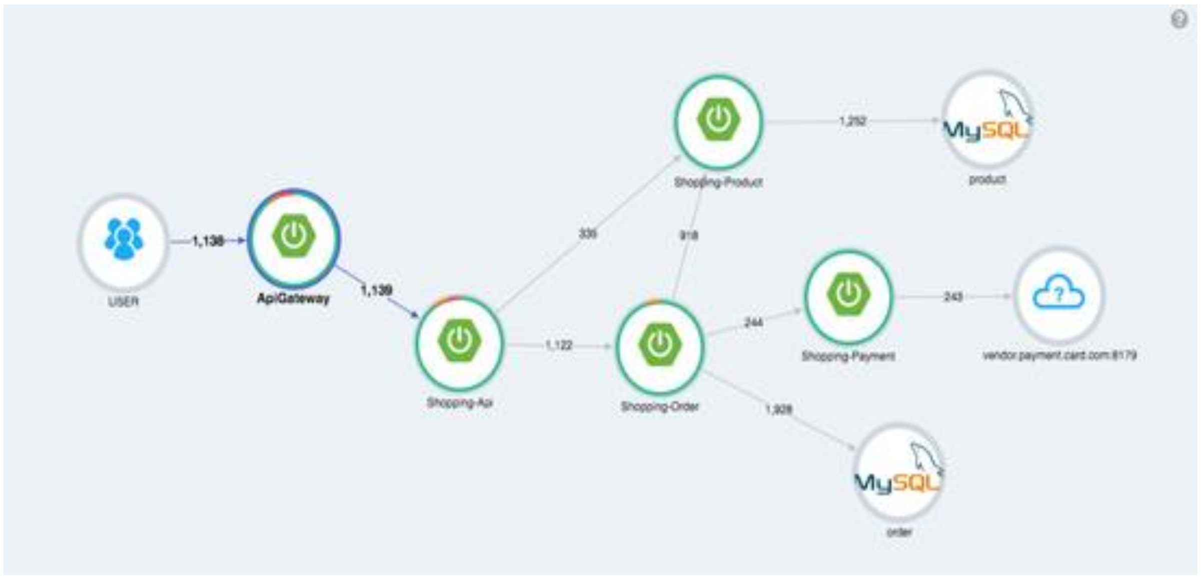


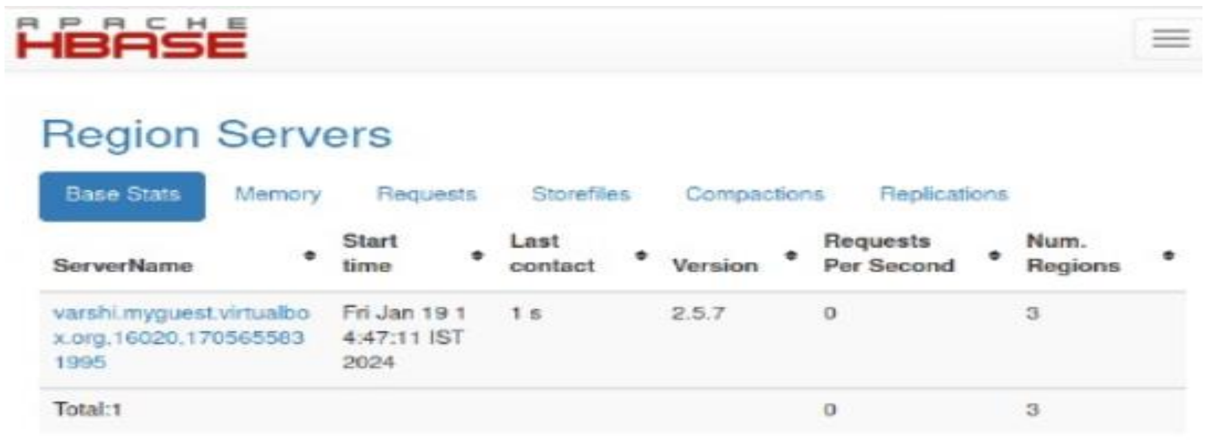
Fig 6.1.1 message pattern

ServerMap provides a visual representation of distributed system components and their interconnections, offering insights into current status and transaction counts. Realtime Active Thread Chart tracks active threads, while Request/Response Scatter Chart visualizes request/response patterns for issue identification.

HBASE CALCULATIONS:

Total storage capacity = (Number of Regions) × (Average Region size)

Throughput = (Number of read/write operations per second) / (Response time per operation)



Region Servers

ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
varshi.myguest.virtualbox.org,16020,1705655831995	Fri Jan 19 14:47:11 IST 2024	1 s	2.5.7	0	3
Total:1				0	3

Fig 6.1.2 hbase calculations

LOGS FORMULA:

Log Volume per Hour:

$$\text{Log volume per hour} = \frac{\text{Total number of logs}}{\text{Time period}}$$

$$\text{Log volume per hour} = \frac{1000\text{ logs}}{1\text{ hour}} = 1000\text{ logs/hour}$$

Fig 6.1.3 logs volume per hour

Log volume per hour is a metric used to quantify the volume of log entries captured within a specific timeframe, typically one hour.

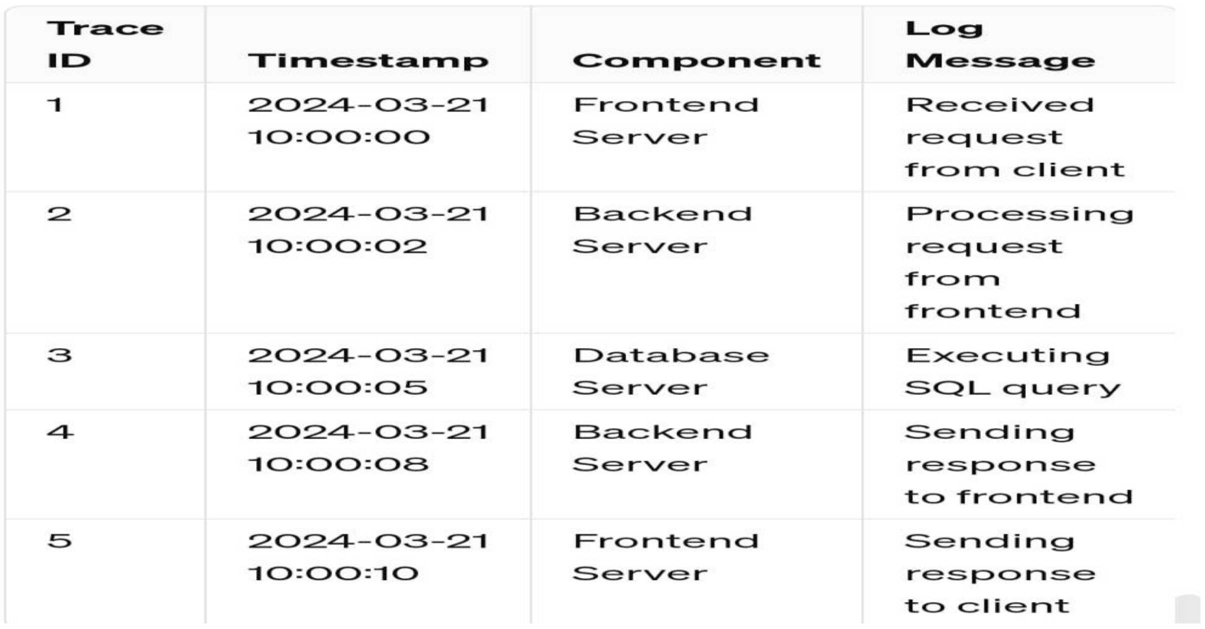
Error Rate:

$$\text{Error rate} = \left(\frac{\text{Number of error logs}}{\text{Total number of logs}}\right) \times 100\%$$

$$\text{Error rate} = \left(\frac{50}{1000}\right) \times 100\% = 5\%$$

Fig 6.1.4 error rate

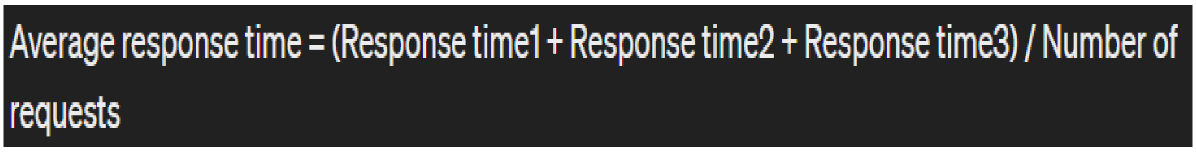
Error rate quantifies the proportion of error logs compared to the total log count, expressed as a percentage. Count error logs and total logs, then divide errors by total logs and multiply by 100% to obtain the error rate.



Trace ID	Timestamp	Component	Log Message
1	2024-03-21 10:00:00	Frontend Server	Received request from client
2	2024-03-21 10:00:02	Backend Server	Processing request from frontend
3	2024-03-21 10:00:05	Database Server	Executing SQL query
4	2024-03-21 10:00:08	Backend Server	Sending response to frontend
5	2024-03-21 10:00:10	Frontend Server	Sending response to client

Fig 6.1.5 logs table

TRACES AND METRICS FORMULA:



Average response time = (Response time1 + Response time2 + Response time3) / Number of requests

Fig 6.1.6 average response time

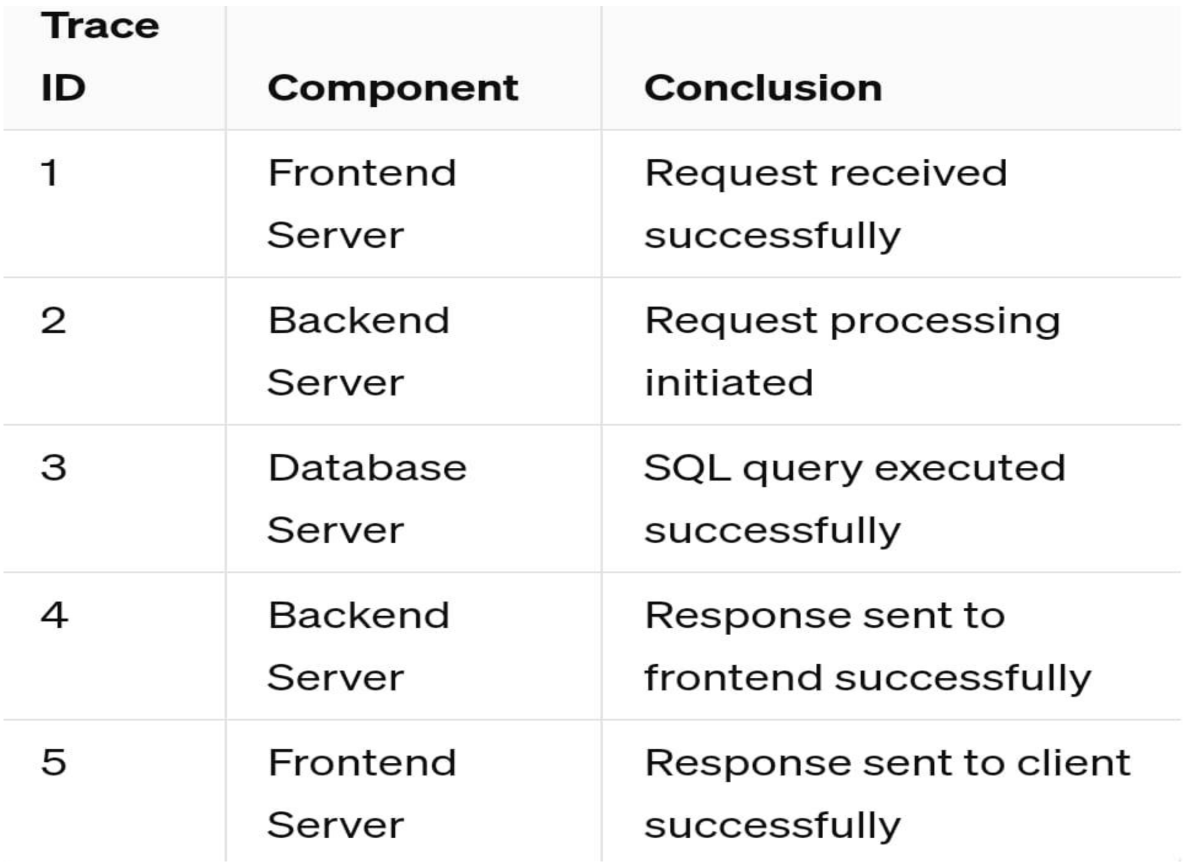
Frontend - Request received: The client initiates the request, and the server's frontend receives it successfully.

Backend - Request processing: The request is handed off to the server's backend for processing.

Database - SQL query executed: The backend executes a SQL query on the database server, successfully.

Backend - Response sent to frontend: The backend prepares a response to send to the frontend.

Frontend - Response sent to client: The frontend receives the response from the backend and successfully sends it to the client.



Trace ID	Component	Conclusion
1	Frontend Server	Request received successfully
2	Backend Server	Request processing initiated
3	Database Server	SQL query executed successfully
4	Backend Server	Response sent to frontend successfully
5	Frontend Server	Response sent to client successfully

Fig 6.1.7 conclusion matrix

PERFORMANCE METRICS: Pinpoint Web displays performance metrics, such as response times, error rates, and throughput, for each transaction. These metrics help in identifying performance bottlenecks and areas for optimization.

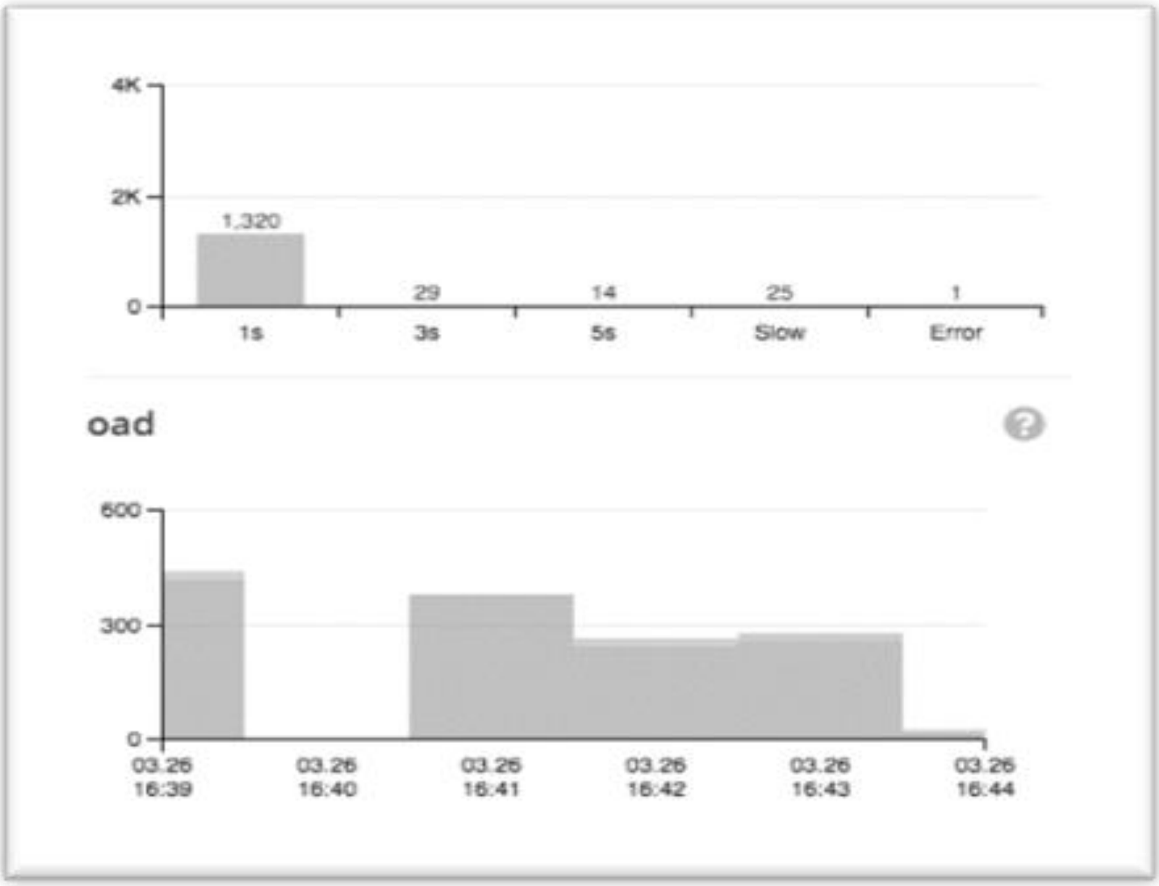


Fig 6.1.8 load and metrics

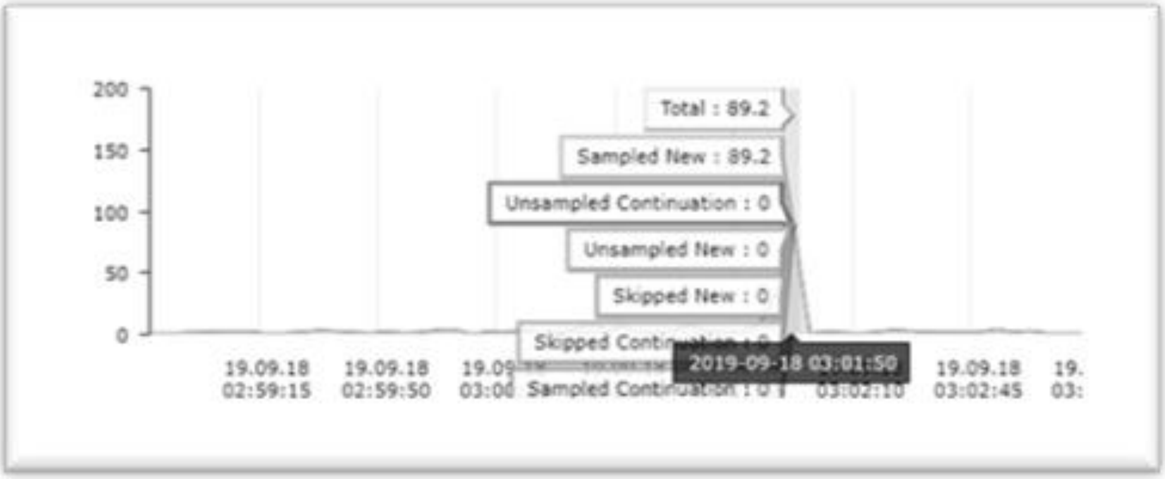


Fig 6.1.9 Graph of load traces and metrics:

6.2 RESULTS AND DISCUSSION:

Pinpoint offers a robust solution for root cause analysis, problem resolution, and continuous monitoring and optimization of system performance. By identifying the exact components and services involved in slow or failing transactions, Pinpoint facilitates targeted root cause analysis, empowering developers to address performance issues promptly. Armed with insights gleaned from Pinpoint, developers can optimize resource utilization, enhance system reliability, and iteratively implement architectural improvements. Continuous monitoring provided by Pinpoint enables developers to track performance trends in real-time and respond promptly to any deviations. By integrating Pinpoint Agents into distributed applications, developers can capture transaction data, including method invocations, database queries, and remote service calls. This instrumentation allows for comprehensive monitoring and analysis of application performance. Insights derived from Pinpoint enable developers to fine-tune resource allocation, optimize application performance, and implement iterative improvements to enhance system efficiency and reliability. Overall, Pinpoint serves as a valuable tool for maintaining optimal system performance and ensuring a seamless user experience.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Monitoring is a term frequently used across many domains in IT. In order to narrow down the exact meaning, we did a literature research to identify a widely accepted definition of the term in the area of system monitoring. We advanced the research and revisited further terminology in the area of monitoring including instrumentation and observability. We further broke down the process of monitoring into several individual functional blocks (acquisition, processing, storage, analysis, reporting) and define an EBF for their composition. Subsequently, we analyse the architectures of IoT systems in order to capture the current state of the art in that field. We extrapolate this architecture to the concepts of the rising Cloud to-Thing-Continuum. The operational layer range from the physical layer over multiple virtualization layers to the application layer, while the realms range from IoT-Devices towards the core network and to cloud data centres. Combining the conclusions from all previous research questions, we build up requirements towards a wide-area monitoring system for the Cloud-to-Thing Continuum. Finally, we evaluate whether state of the art monitoring tools match the brought up requirements and hence are suited for holistic and flexible monitoring of applications and infrastructures in the Cloud-to-Thing-Continuum. The outcome of that analysis is that most requirements are satisfied by some tool, but as of today there is not holistic or interoperable solution available.

7.2 FUTURE ENHANCEMENT

The answering our core research questions, further subsequent questions arise. The following summarizes these questions, their inherent challenges and approaches towards their respective solutions or answers.

An important question is the quantification of observability. According to our definition, monitoring is a means to achieve observability.

However, currently, not direct or even indirect metric exists to quantify the state of observability of a system. Future work within this scope should create such a quantification metric.

This involves answering the question how to measure observability and how degrees of observability can be compared.

Furthermore building upon our results detailed analysis regarding the fulfilment of our requirements by monitoring tools should be made including research papers.

This step involves the creation of an observability landscape based on our requirements and definitions.

Finally, a methodology as well as testbeds needs to be developed that allows to evaluated the capabilities of monitoring tools with respect to the Cloud-to-Thing-Continuum on a large scale.

APPENDICES

A1 SDG GOALS

Industry, Innovation, and Infrastructure(SDG-9):

Implementing effective instrumentation and monitoring systems

contributes to the development of sustainable infrastructure and promotes innovation in technology and industry.

Sustainable Cities and Communities(SDG-11):

Monitoring virtual systems helps in creating sustainable and resilient cities and communities by ensuring the efficient use of resources, improving infrastructure reliability, and enhancing disaster preparedness.

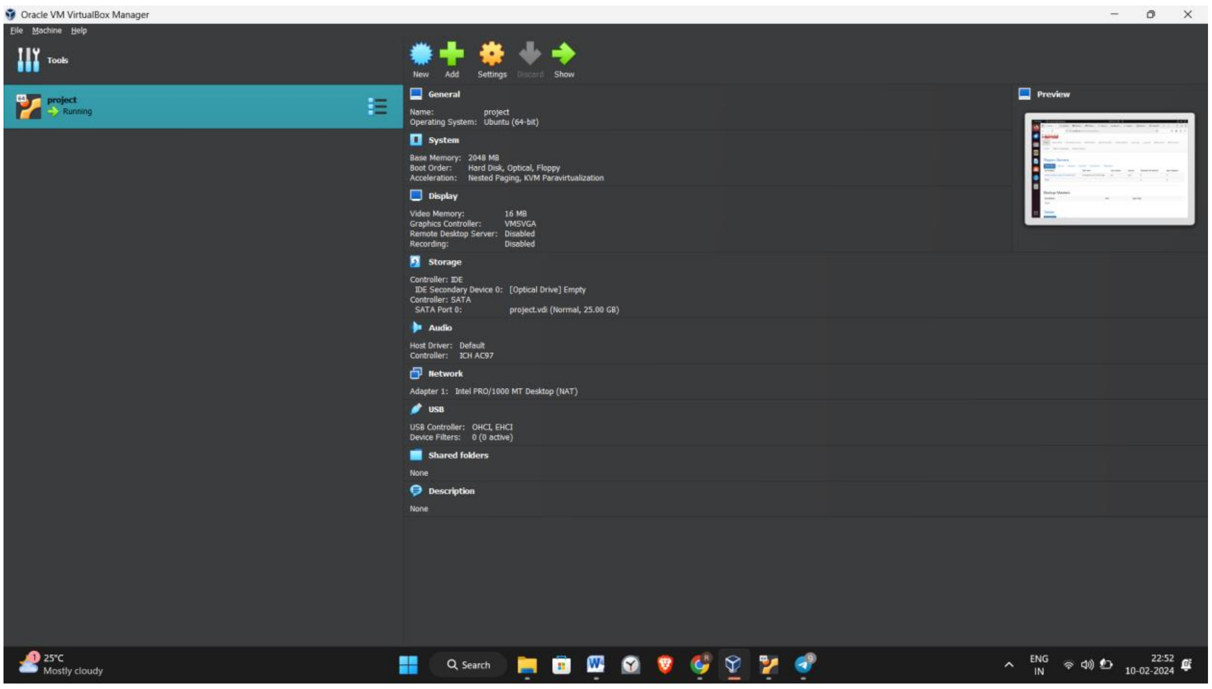
Climate Action(SDG 13):

Effective monitoring of virtual systems can help reduce energy consumption, optimize resource usage, and minimize carbon emissions, thus contributing to efforts to mitigate climate change.

Life on Land(SDG-15):

Monitoring virtual systems can aid in the conservation and sustainable management of land resources by providing insights into environmental impacts and facilitating land-use planning and conservation efforts.

A2 SAMPLE SCREENSHOTS



project
Running

General
Name: project
Operating System: Ubuntu (64-bit)

System
Base Memory: 2048 MB
Boot Order: Hard Disk, Optical, Floppy
Acceleration: Nested Paging, KVM Paravirtualization

Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: IDE
IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
SATA Port 0: project.vdi (Normal, 25.00 GB)

Audio
Host Driver: Default
Controller: ICH AC97

Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)

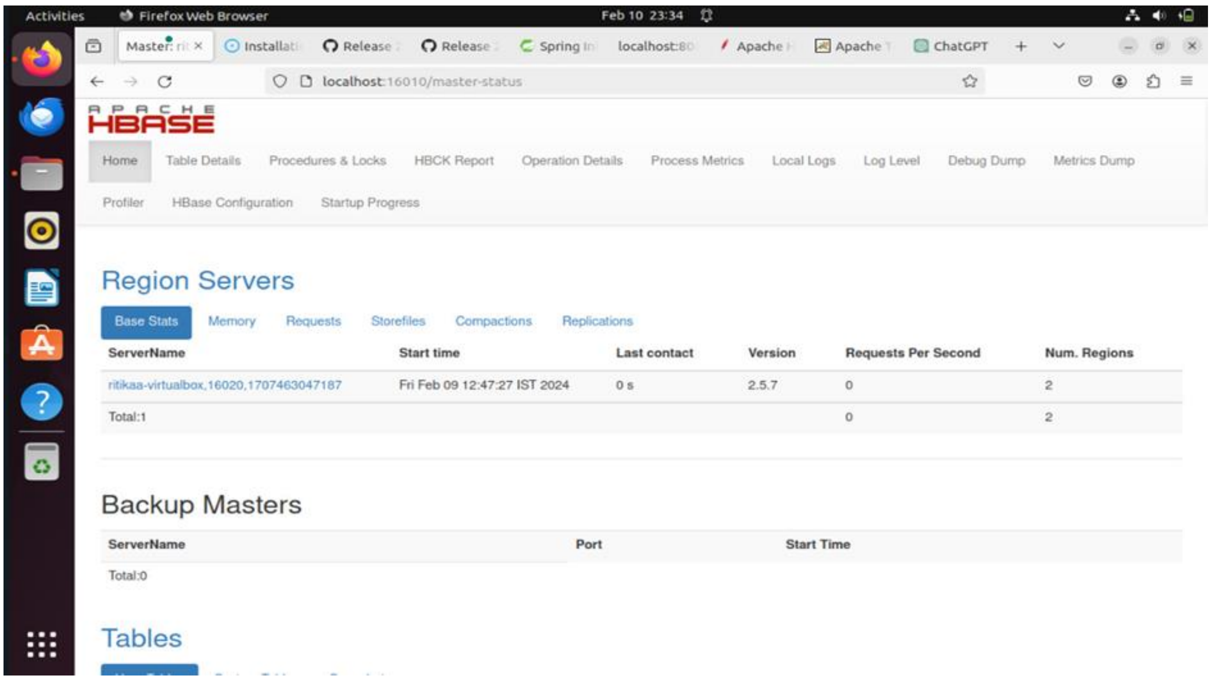
USB
USB Controller: OHCI, EHCI
Device Filters: 0 (0 active)

Shared folders
None

Description
None

25°C Mostly cloudy Search ENG IN 22:52 10-02-2024

Fig 8.2.1 creating a virtual environment



Region Servers

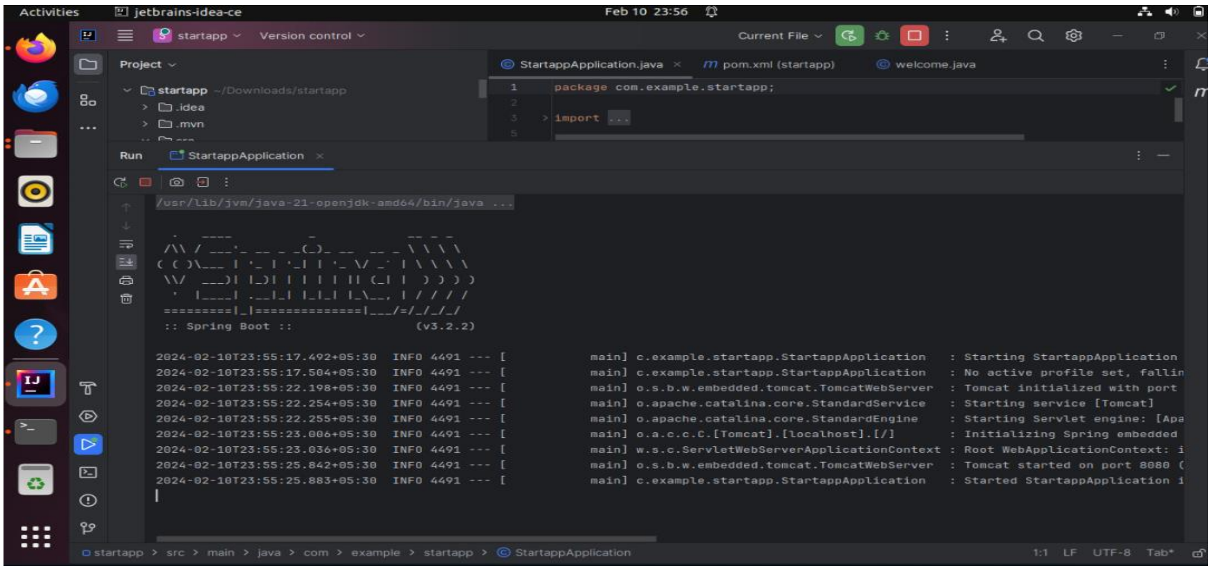
ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
ritikaa-virtualbox,16020,1707463047187	Fri Feb 09 12:47:27 IST 2024	0 s	2.5.7	0	2
Total:1				0	2

Backup Masters

ServerName	Port	Start Time
Total:0		

Tables

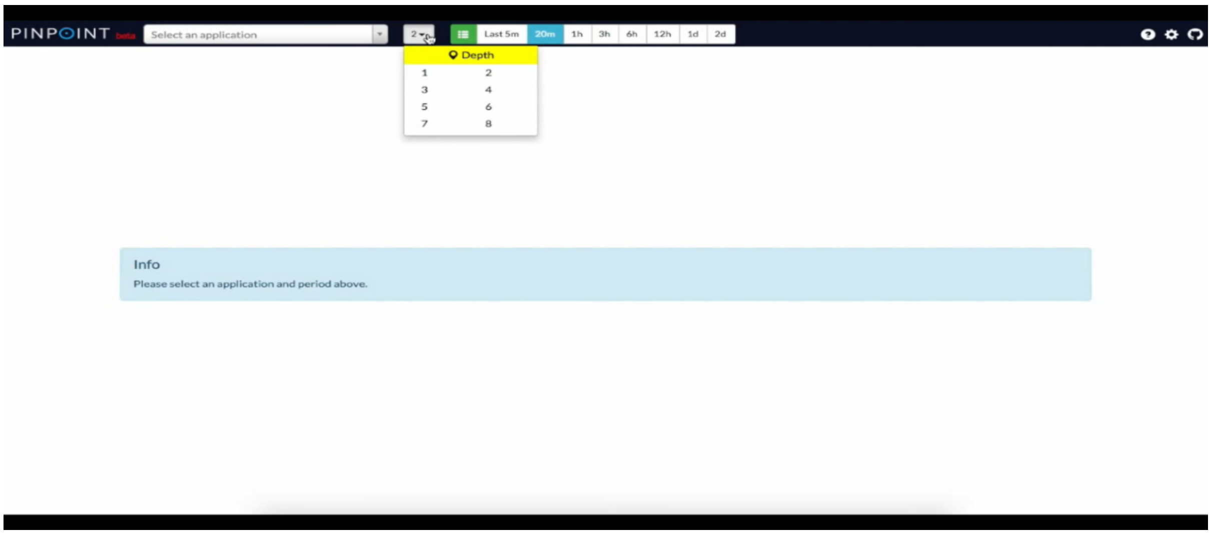
Fig 8.2.2 created a HBase table



```
package com.example.startapp;

import ...
```

```
/usr/lib/jvm/java-21-openjdk-amd64/bin/java ...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.2)

2024-02-10T23:55:17.492+05:30  INFO 4491 --- [           main] c.example.startapp.StartappApplication   : Starting StartappApplication
2024-02-10T23:55:17.504+05:30  INFO 4491 --- [           main] c.example.startapp.StartappApplication   : No active profile set, fallin
2024-02-10T23:55:22.198+05:30  INFO 4491 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port
2024-02-10T23:55:22.254+05:30  INFO 4491 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-02-10T23:55:22.255+05:30  INFO 4491 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apa
2024-02-10T23:55:23.006+05:30  INFO 4491 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
2024-02-10T23:55:23.036+05:30  INFO 4491 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: i
2024-02-10T23:55:25.842+05:30  INFO 4491 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (
2024-02-10T23:55:25.883+05:30  INFO 4491 --- [           main] c.example.startapp.StartappApplication   : Started StartappApplication i
```

Fig 8.2.3 created a spring boot application



Info
Please select an application and period above.

Fig 8.2.4 pinpoint setup

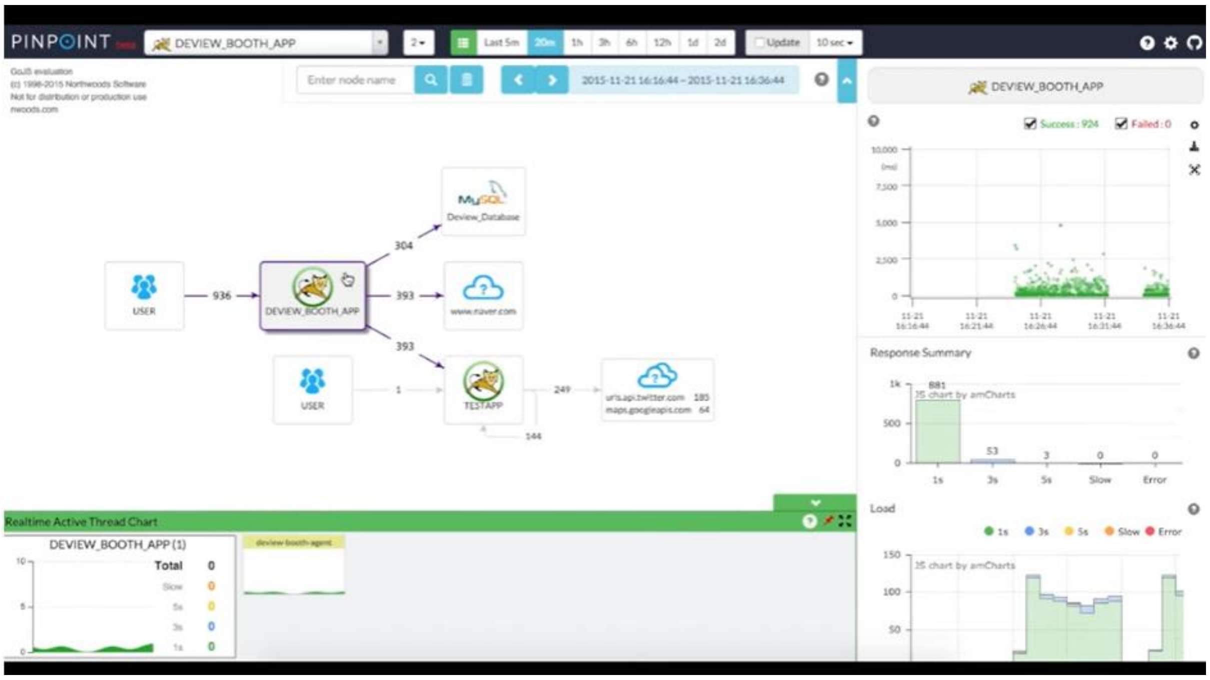
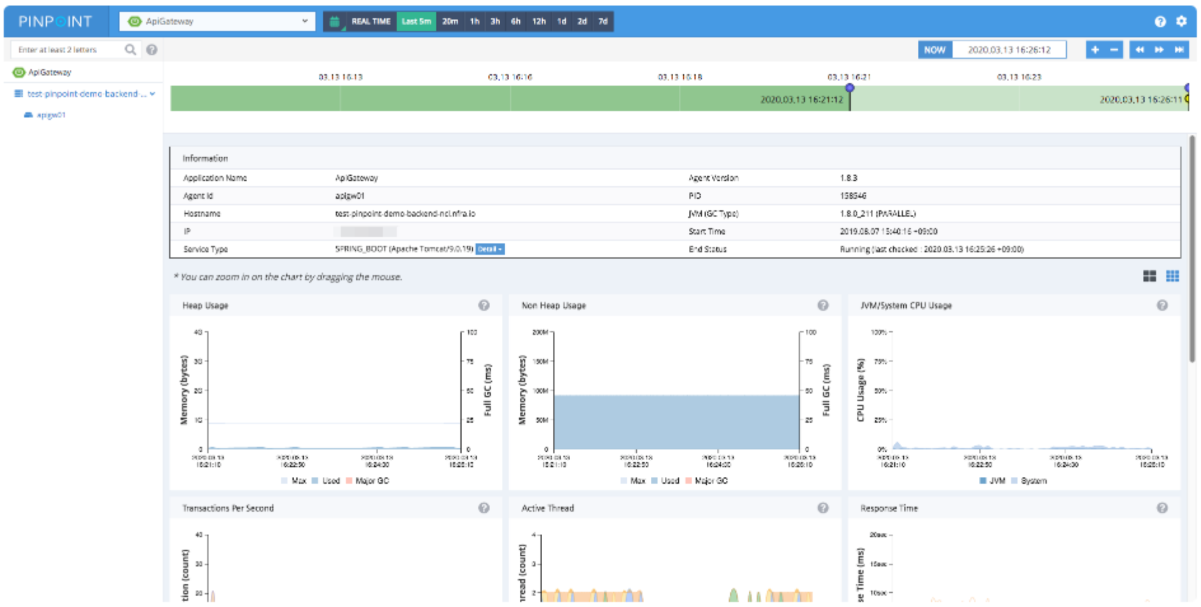


Fig 8.2.5 server map

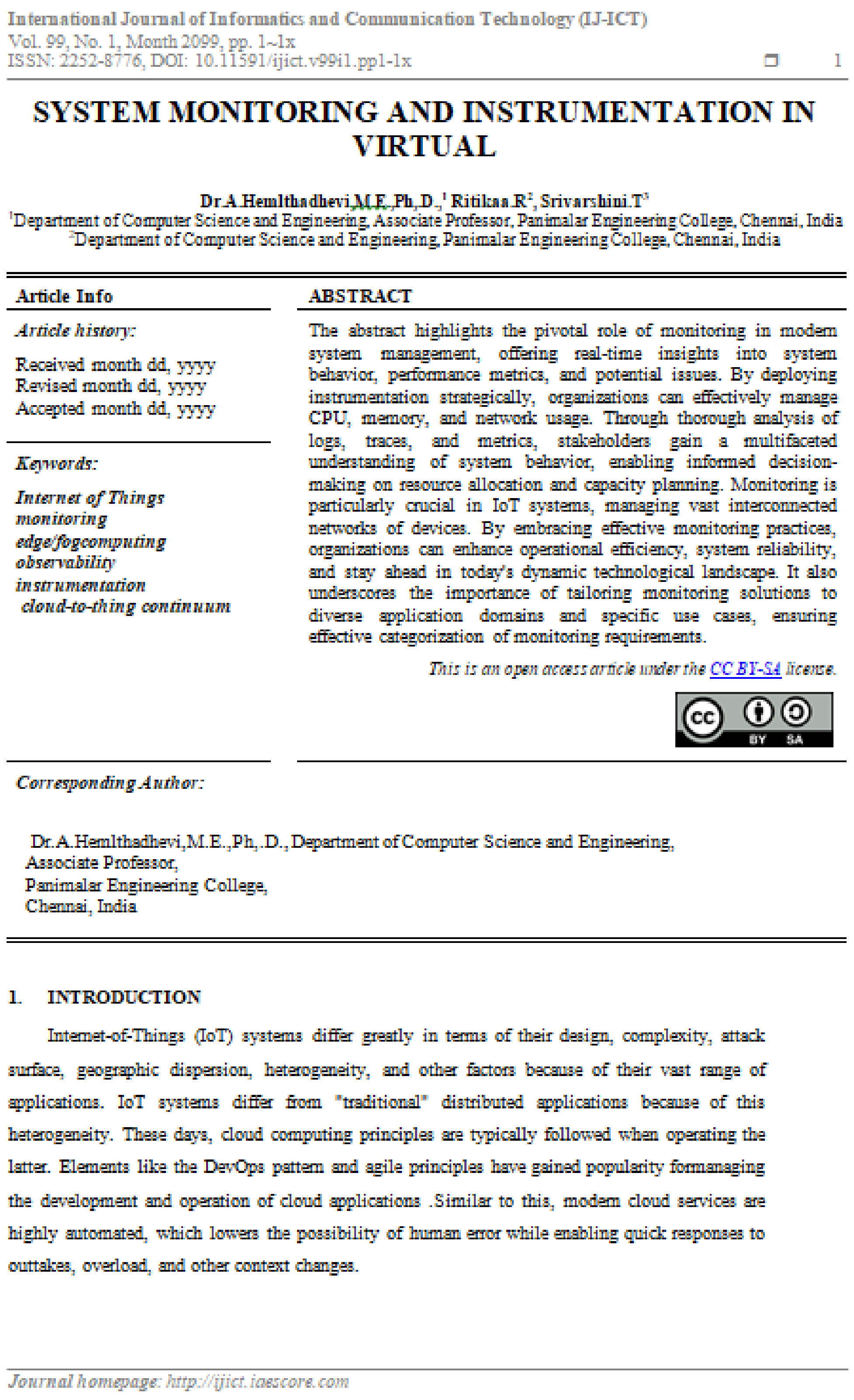


Information			
Application Name	ApiGateway	Agent Version	1.8.3
Agent Id	apigw01	PID	158546
Hostname	test-pinpoint-demo-backend-ncl.nfra.io	JVM (GC Type)	1.8.0_211 (PARALLEL)
IP	[redacted]	Start Time	2019.08.07 15:40:16 +09:00
Service Type	SPRING_BOOT (Apache Tomcat/9.0.19) Detail	End Status	Running (last checked : 2020.03.13 16:25:26 +09:00)

* You can zoom in on the chart by dragging the mouse.

Fig 8.2.6 Inspector

A3. PAPER PUBLICATION



International Journal of Informatics and Communication Technology (IJ-ICT)
Vol. 99, No. 1, Month 2099, pp. 1~1x
ISSN: 2252-8776, DOI: 10.11591/ijict.v99i1.pp1-1x ❐ 1

SYSTEM MONITORING AND INSTRUMENTATION IN VIRTUAL

Dr.A.Hemlthadhevi,M.E.,Ph,.D.,[1] Ritikaa.R[2], Srivarshini.T[3]
[1]Department of Computer Science and Engineering, Associate Professor, Panimalar Engineering College, Chennai, India
[2]Department of Computer Science and Engineering, Panimalar Engineering College, Chennai, India

Article Info

Article history:

Received month dd, yyyy
Revised month dd, yyyy
Accepted month dd, yyyy

Keywords:

Internet of Things
monitoring
edge/fogcomputing
observability
instrumentation
cloud-to-thing continuum

ABSTRACT

The abstract highlights the pivotal role of monitoring in modern system management, offering real-time insights into system behavior, performance metrics, and potential issues. By deploying instrumentation strategically, organizations can effectively manage CPU, memory, and network usage. Through thorough analysis of logs, traces, and metrics, stakeholders gain a multifaceted understanding of system behavior, enabling informed decision-making on resource allocation and capacity planning. Monitoring is particularly crucial in IoT systems, managing vast interconnected networks of devices. By embracing effective monitoring practices, organizations can enhance operational efficiency, system reliability, and stay ahead in today's dynamic technological landscape. It also underscores the importance of tailoring monitoring solutions to diverse application domains and specific use cases, ensuring effective categorization of monitoring requirements.

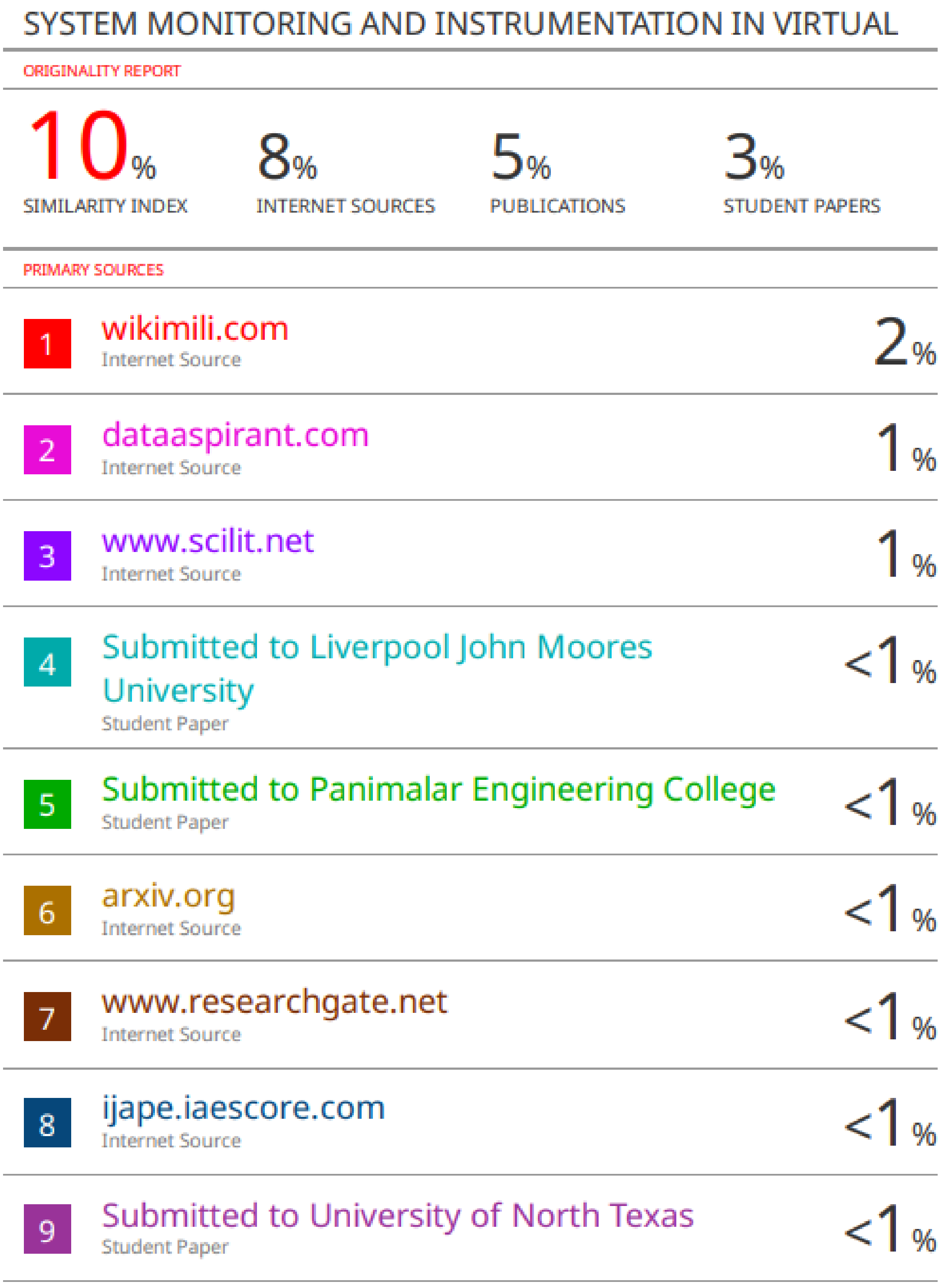
Corresponding Author:

Dr.A.Hemlthadhevi,M.E.,Ph,.D., Department of Computer Science and Engineering,
Associate Professor,
Panimalar Engineering College,
Chennai, India

1. INTRODUCTION

Internet-of-Things (IoT) systems differ greatly in terms of their design, complexity, attack surface, geographic dispersion, heterogeneity, and other factors because of their vast range of applications. IoT systems differ from "traditional" distributed applications because of this heterogeneity. These days, cloud computing principles are typically followed when operating the latter. Elements like the DevOps pattern and agile principles have gained popularity formanaging the development and operation of cloud applications .Similar to this, modern cloud services are highly automated, which lowers the possibility of human error while enabling quick responses to outtakes, overload, and other context changes.

A4. PLAGIARISM REPORT



SYSTEM MONITORING AND INSTRUMENTATION IN VIRTUAL

ORIGINALITY REPORT

10%	8%	5%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	wikimili.com Internet Source	2%
2	dataaspirant.com Internet Source	1%
3	www.scilit.net Internet Source	1%
4	Submitted to Liverpool John Moores University Student Paper	<1%
5	Submitted to Panimalar Engineering College Student Paper	<1%
6	arxiv.org Internet Source	<1%
7	www.researchgate.net Internet Source	<1%
8	ijape.iaescore.com Internet Source	<1%
9	Submitted to University of North Texas Student Paper	<1%

REFERENCES

[1] M.F. Lie, M. Sánchez-Gordón, R. Colomo-Palacios, DevOps in an ISO 13485 regulated environment, in Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), ACM, 2020, http://dx.doi.org/10.1145/3382494.3410679.

[2] M. Rajkumar, A.K. Pole, V.S. Adige, P. Mahanta, DevOps culture and its impact on cloud delivery and software development, in 2016 International Conference on Advances in Computing, Communication, Automation (ICACCA) (Spring), 2016, pp. 1–6, http://dx.doi.org/10.1109/ICACCA.2016. 7578902.

[3] M. Abderrahim, M. Ouzzif, K. Guillouard, J. Francois, A. Lebre, A holistic monitoring service for fog/edge infrastructures: A foresight study, in 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), 2017, pp. 337–344, http://dx.doi.org/10.1109/FiCloud.2017.30.

[4] C. Tsigkanos, S. Nastic, S. Dustdar, Towards resilient internet of things: Vision, challenges, and research roadmap, in 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, 2019, pp. 1754–1764, http://dx.doi.org/10.1109/ICDCS.2019.00174.

[5] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990 (1990) 1–84, http://dx.doi.org/10.1109/IEEESTD.1990.101064.

[6] C. Berger, P. Eichhammer, H.P. Reiser, J. Domaschka, F.J. Hauck, G. Habiger, A survey on resilience in the IoT: Taxonomy, classification, and

discussion of resilience mechanism, ACM Comput. Surv. (2021) accepted for publication. [7] G. Lee, B.-G. Chun, R.H. Katz, Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud 5.

[7] M. Iorga, L. Feldman, R. Barton, M. Martin, N. Goren, C. Mahmoudi, The NIST definition of fog computing, Tech. rep., National Institute of Standards and Technology, 2017.

[8] V. Karagiannis, S. Schulte, Comparison of alternative architectures in fog computing, in 2020 IEEE 4th International Conference on Fog and Edge Computing, ICFEC, IEEE, 2020, pp. 19–28.

[9] S. Taheri Zadeh, A.C. Jones, I. Taylor, Z. Zhao, V. Stankovski, Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review, J. Syst. Softw. 136 (2018) 19–38, http://dx.doi.org/10.1016/ j.jss.2017.10.033.

[10] O. Bello, S. Zeadally, M. Badra, Network layer inter-operation of device-to-device communication technologies in internet of things (IoT), Ad Hoc Netw. 57 (C) (2017) 52–62, http://dx.doi.org/10.1016/j.adhoc.2016.06.010.

[11] J.S. Ward, A. Barker, Observing the clouds: a survey and taxonomy of cloud monitoring, J. Cloud Comput. 3 (1) (2014) 1–30. [13] J. Joyce, G. Lomow, K. Slind, B. Unger, Monitoring distributed systems, ACM Trans. Comput. Syst. 5 (2) (1987) 121–150,http://dx.doi.org/10.1145/ 13677.22723.

[12] J.S. Ward, A. Barker, Observing the clouds: a survey and taxonomy of cloud monitoring, J. Cloud Comput.3 (1) (2014) 1–30.

[13] J. Joyce, G. Lomow, K. Slind, B. Unger, Monitoring distributed systems, ACMTrans.Comput.Syst.5(2)(1987)121,150,http://dx.doi.org/10.1145/13677.22723.