UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Internet of Things Project

Smart Datacenter IoT Telemetry and Control System

Student:

Francesco Londretti

YEAR 2022/2023

Contents

Chapter 1

Introduction

The Smart Datacenter IoT Telemetry and Control System is a cutting-edge project designed to monitor and optimize the operations of a datacenter using Internet of Things (IoT) technology. By deploying a network of sensors and actuators, the system collects real-time data on various parameters within the datacenter environment.

With the use of sensors, such as temperature and current sensors, the system continuously monitors crucial metrics. This data is then processed and stored in a database, enabling advanced analytics and insights. Based on the collected data, the system employs actuators to implement intelligent control actions, such as adjusting cooling systems or optimizing energy consumption.

The user has the ability to customize and define control logic through a terminal interface, empowering them to tailor the system to specific requirements. Additionally, a web-based interface provides intuitive data visualization, allowing users to monitor and analyze the collected data in real-time.

The Smart Datacenter IoT Telemetry and Control System offers numerous benefits, including improved operational efficiency, energy optimization, and proactive maintenance. By leveraging IoT technologies, it enables data-driven decision-making and enhances the overall management of datacenter infrastructure.

Chapter 2

System Architecture

The datacenter monitoring system consists of multiple components that work together to monitor and control the datacenter environment. The main components include:

- Sensors: Simulated temperature and power sensors are deployed within the datacenter to collect real-time data. MQTT protocol is used for communication between the sensors and the collector application.

- Actuators: Simulated actuators, such as water cooling and power modules, are used to control the temperature and power consumption within the datacenter. CoAP protocol is used for communication between the actuators and the collector application.

- Collector Application: A Java-based collector application acts as the central control unit. It collects data from sensors, orchestrates the operation of actuators based on the received data, and provides a user interface for monitoring and configuration.

The system architecture follows a distributed model, where sensors are deployed at various locations within the datacenter, and the collector application serves as the central hub for data aggregation and control.

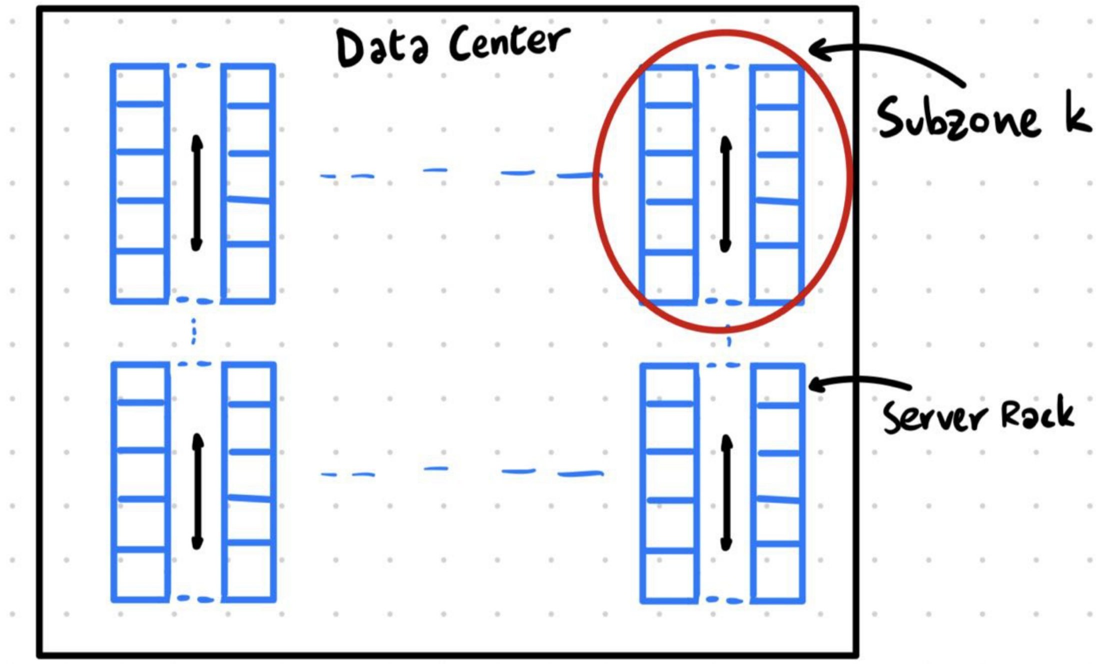

Figure 2.1: Sample structure of the datacenter

2.1 Sensor Simulation

The system simulates two types of sensors: temperature and power sensors. The temperature sensor measures the ambient temperature within the datacenter, while the power sensor measures the consumed power. These sensors are implemented in C programming language within the Contiki operating system.

The temperature sensor periodically reads the temperature and adjusts it based on the status of the water cooling system. If the water cooling system is active, it decreases the temperature by a random amount; otherwise, it decreases the temperature by a smaller random amount. The temperature value is bounded within a specified range.

The power sensor simulates the consumed power and adjusts it randomly. If the datacenter is in an overload state, it reduces the power consumption to alleviate the workload. The power value is also bounded within a specified range.

MQTT protocol is used for communication between the sensors and the collector application. The sensors publish temperature and power data to specific topics, and the collector application subscribes to these topics to receive the data.

2.2 Actuator Control

The system includes two types of actuators: water cooling and power modules. The water cooling actuator controls the cooling mechanism within the datacenter, while the power module regulates the power consumption.

The water cooling actuator can be turned on or off. When it is active, it cools the datacenter by reducing the temperature. The power module can also be turned on or off. If the power module is off, the subzone associated with it is turned off to save energy.

CoAP protocol is used for communication between the actuators and the collector application. The collector application sends CoAP requests to the actuators to control their operation. For example, to turn on the water cooling actuator, the collector application sends a CoAP request to the corresponding resource exposed by the actuator.

2.3 LED and Button Integration

To enhance the monitoring and control capabilities of the datacenter monitoring system, LED indicators and buttons are integrated into the implementation.

The LEDs are used to provide visual feedback and status indication.

The green LED on the sensors blinks periodically to indicate that the sensor is actively sending data.

In the power actuator, the LED is blue when the power is on and there are no issues. However, if the subzone is in an overload state, the LED alternates between blue and green, indicating an emergency situation in that area.

In the cooling actuator, the LED is blue when the fan is active. If the water cooling is active the LED is green instead.

Buttons are used to initiate the operation of the system. When a button is pressed, the corresponding functionality of the system is activated. For example, pressing a button may trigger the initialization of the sensors or actuators, establish the connection with the collector application, or start a specific monitoring or control process.

The integration of LEDs and buttons enhances the user experience and provides a convenient interface for system operation and status monitoring.

2.4 Collector Application

The collector application serves as the orchestrator of the system, collecting data from the sensors and making decisions for actuator control. It runs on a Java platform and communicates with the sensors and actuators via CoAP (Constrained Application Protocol).

The collector application subscribes to the sensor topics and receives the temperature and power data from each subzone. It analyzes the collected data and determines the appropriate actions for the actuators based on predefined temperature and power bounds.

The collector application offers a user-friendly interface with various functionalities. Users can query the current temperature and power readings of each subzone using the `get_temp` and `get_pow` commands, respectively. They can also modify the temperature and power bounds of specific subzones using the `set_temp` and `set_pow_bounds` commands. Additionally, an `overview` command provides a comprehensive view of the temperature, power, and state information for all subzones.

By centralizing the data collection and decision-making process, the collector application enables efficient monitoring and control of the datacenter. It empowers users to make informed decisions, optimize resource allocation, and respond promptly to critical situations.

Data Format: JSON vs. XML

Considering the simplicity, compatibility, and flexibility of JSON, it emerged as the preferred data format for the datacenter monitoring system. It allows for efficient data transmission, ease of implementation, and seamless integration with the collector application.

Chapter 3

Web-based Interface and Data Visualization

The web-based interface plays a crucial role in the datacenter monitoring system by providing real-time data visualization and user-friendly control functionalities. This chapter focuses on the development of the interface using Grafana, how it connects to the database, and its powerful data visualization capabilities.

3.1 Database Structure

The datacenter monitoring system utilizes a MySQL database to store the collected data. The database is designed with the following structure:

Table 3.1: Table: consumed_power

Field	**Type**	**Null**	**Key**	**Extra**
subzone	int	YES		
value	int	YES		
timestamp	timestamp	YES		CURRENT_TIMESTAMP

This table stores information about the consumed power in different subzones of the datacenter. It has the following columns:

- ‘subzone‘: An integer field that represents the subzone in the datacenter. - ‘value‘: An integer field that stores the consumed power value in the corresponding subzone. - ‘timestamp‘: A timestamp field that records the date and time when the power consumption data was recorded. It has a default value of the current timestamp.

Table 3.2: Table: temperature

Field	Type	Null	Key	Extra
subzone	int	YES		
value	float	YES		
timestamp	timestamp	YES		CURRENT_TIMESTAMP

This table stores information about the temperature measurements in different subzones of the datacenter. It has the following columns:

- ‘subzone‘: An integer field that represents the subzone in the datacenter. - ‘value‘: A floating-point field that stores the temperature value recorded in the corresponding subzone. - ‘timestamp‘: A timestamp field that records the date and time when the temperature measurement was taken. It has a default value of the current timestamp.

These tables serve as the main storage for the collected data from the sensors in the datacenter. The ‘consumed_power‘ table tracks the consumed power in various subzones, while the ‘temperature‘ table records the temperature measurements. The timestamp column in both tables allows for tracking the time of each data entry, enabling historical analysis and real-time monitoring.

3.2 Data Visualization Capabilities

The web-based interface provides powerful data visualization capabilities through Grafana’s features and plugins. It allows users to monitor the datacenter environment in real-time and gain valuable insights from the collected data.

3.2.1 Real-time Monitoring

Grafana enables real-time monitoring of temperature and power readings from each subzone. By continuously querying the database for the latest data, the interface updates the visualizations in real-time, providing up-to-date information on the current state of the datacenter.

3.2.2 Customizable Dashboards

The interface offers customizable dashboards that allow users to create personalized views of the datacenter monitoring system. I have created four views within the Grafana dashboard:

1. **Temperature of a Subzone:** This view displays the temperature readings of a selected subzone. It includes a line chart that visualizes the temperature over time, allowing users to track temperature fluctuations and identify any anomalies. Users can choose the subzone to view using a variable, which provides a dropdown list of available subzones.

2. **Consumed Power of a Subzone:** Similar to the temperature view, this panel presents the consumed power readings of a selected subzone. It utilizes a line chart to visualize the power consumption over time, enabling users to monitor power usage patterns and detect irregularities.

3. **Average Temperature of All Subzones:** This view provides an overview of the average temperature across all subzones. It presents a single value visualization that displays the current average temperature, allowing users to quickly assess the overall temperature condition of the datacenter.

4. **Average Consumed Power of All Subzones:** Similarly, this panel showcases the average consumed power across all subzones. It employs a gauge visualization that represents the current average power consumption, giving users an instant understanding of the datacenter's power usage efficiency.

These views offer a comprehensive monitoring experience, providing users with the ability to focus on specific subzones or obtain an overall perspective of the datacenter environment.

3.2.3 Queries and Variables

To populate the visualizations, the web-based interface utilizes SQL queries to fetch data from the database. These queries retrieve the temperature and power readings based on the selected subzone and time range.

```sql
SELECT value, timestamp
FROM ioT_Project.temperature
WHERE subzone = $subzone
ORDER BY timestamp ASC
```

Listing 3.1: MySQL code for retrieving the temperature readings.

```sql
SELECT value, timestamp
FROM ioT_Project.`consumed power`
WHERE subzone = $subzone
ORDER BY timestamp ASC
```

Listing 3.2: MySQL code for retrieving the consumed power readings.

```sql
SELECT subzone, AVG(value) AS "Average Temperature (degrees C)
    "
FROM ioT_Project.temperature
GROUP BY subzone
ORDER BY subzone
```

Listing 3.3: MySQL code for the average temperature readings for each subzone.

```sql
SELECT subzone, AVG(value) / 1000 AS "Average Consumed Power (
    kW)"
FROM ioT_Project.`consumed power`
GROUP BY subzone
ORDER BY subzone
```

Listing 3.4: MySQL code for the average consumed power readings for each subzone.

The interface also incorporates '$subzone' a variable that allows users to choose the subzone they wish to monitor. This variable dynamically updates the queries, ensuring that the visualizations display data specific to the selected subzone.

```sql
SELECT DISTINCT subzone
FROM temperature;
```

Listing 3.5: MySQL code for the variable *subzone*

Chapter 4

Conclusions

In conclusion, the datacenter monitoring system presented in this report offers a comprehensive solution for real-time data collection, analysis, and control. By utilizing simulated sensors and actuators, the system provides an efficient means of monitoring temperature and power consumption within the datacenter environment.

While the current implementation focuses on temperature and power monitoring, there is potential for further improvement by incorporating additional sensors. For example, the integration of humidity sensors could provide valuable insights into the datacenter's humidity levels, allowing for better environmental control and preventing issues related to excessive moisture or dryness. Additionally, the inclusion of sensors for air quality, air flow, and equipment health could enhance the system's ability to monitor and optimize the overall datacenter conditions.

The division of the datacenter into subzones makes this project particularly interesting and valuable. By monitoring and controlling each subzone individually, the system can precisely identify areas of concern and optimize resource allocation based on specific requirements. This approach allows for targeted responses to critical situations, minimizing disruptions and ensuring the overall stability and efficiency of the datacenter.

In summary, this project serves as a solid foundation for a comprehensive datacenter monitoring system. By expanding the sensor suite to include additional parameters such as humidity, air quality, and equipment health, the system can provide a holistic view of the datacenter environment. The division into subzones allows for precise monitoring and control, optimizing resource allocation and ensuring optimal conditions. With the powerful visualization capabilities of the web-based interface, users can gain valuable insights and make informed decisions to further improve datacenter management and efficiency.